ROBOT MAPPING WITH REAL-TIME INCREMENTAL LOCALIZATION USING EXPECTATION MAXIMIZATION

THESIS

Kevin L. Owens
Captain, USAF

AFIT/GCS/ENG/05-12

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/05-12

ROBOT MAPPING WITH REAL-TIME INCREMENTAL LOCALIZATION USING EXPECTATION MAXIMIZATION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Kevin L. Owens, BGS

Captain, USAF

March 2005

AFIT/GCS/ENG/05-12

ROBOT MAPPING WITH REAL-TIME INCREMENTAL LOCALIZATION USING EXPECTATION MAXIMIZATION

THESIS

Kevin L. Owens, BGS

Captain, USAF

Approved:

/signed/	3 Mar 2005
Dr. Gilbert L. Peterson (Chairman)	Date
/signed/	4 Mar 2005
Dr. Henry B. Potoczny (Member)	Date
/signed/	7 Mar 2005
Dr. John F. Raquet (Member)	Date

Acknowledgements

First and foremost, I extend my profound gratitude and appreciation to my loving wife and family, who have been graciously supportive of this work and all the sacrifices it entailed.

My thanks also go to my wingman, K2, and the rest of the *Friday Lunch Club*, my peers in the AFIT experience, for making this journey fun at times and bearable at others.

Finally, I am grateful to our friends and loved ones who supported me and my family in this endeavor with their prayers, encouragements, and selfless sacrifice.

Kevin L. Owens

Table of Contents

Page

List of Figures

List of Abbreviations

Abbreviation		Page
SLAM	Simultaneous Localization and Mapping	6
CML	Concurrent Mapping and Localization	6
GPS	Global Positioning System	6
HIMM	Histogrammic In-Motion Mapping	11
EM	Expectation Maximization	13
E-step	The expectation step of the EM algorithm	14
M-step	The maximization step of the EM algorithm	14
ASR	Absolute Space Representation	20
ASR	Absolute Space Representation	22
HMM	Hidden Markov Model	42

Abstract

This research effort explores and develops a real-time sonar-based robot mapping and localization algorithm that provides pose correction within the context of a singe room, to be combined with pre-existing global localization techniques, and thus produce a single, well-formed map of an unknown environment. Our algorithm implements an expectation maximization algorithm that is based on the notion of the alpha-beta functions of a Hidden Markov Model. It performs a forward alpha calculation as an integral component of the occupancy grid mapping procedure using local maps in place of a single global map, and a backward beta calculation that considers the prior local map, a limited step that enables real-time processing.

Real-time localization is an extremely difficult task that continues to be the focus of much research in the field, and most advances in localization have been achieved in an off-line context. The results of our research into and implementation of real-time localization showed limited success, generating improved maps in a number of cases, but not all—a trade-off between real-time and off-line processing. However, we believe there is ample room for extension to our approach that promises a more consistently successful real-time localization algorithm.

ROBOT MAPPING WITH REAL-TIME INCREMENTAL LOCALIZATION USING EXPECTATION MAXIMIZATION

I. Introduction

Robotic mapping of physical environments using mobile robots and their sensors is considered one of the most important steps toward achieving truly autonomous robots. There exist fairly robust methods for mapping, but they are all subject to constraints that do not always apply in the real world [44]. One key aspect of robotic mapping is that it has two components: one, sensing the environment and converting that into an internal representation, and two, correcting for the drift that occurs over the course of its travel, illustrated in Figure 1.1. This second component is one of the most difficult aspects of robot mapping, especially when attempted in a real-time context.

This research focuses on the particular task of performing real-time localization, exploring and extending techniques that are based on recent research in the field of robotic mapping.

1.1 Rationale

Deeply buried, hardened facilities have become a promising and considerably effective defense for rogue states and terrorist organizations against US firepower that has otherwise proven itself against surface forces and facilities, and even some buried targets, with extreme precision and lethality. These deeply buried facilities pose numerous challenges to the US military, foremost of which may be the ability to determine their layout and contents, intelligence that is critical to effective targeting and weapon system configuration. Robot technology can assist in this effort.

This work envisions the scenario where sonar-guided robots are deployed through air shafts or tunnels into possibly unmapped, yet structured, deeply buried facilities (i.e., with walls, corridors, and rooms) wherein they conduct real-time mapping, localization, and exploration. This

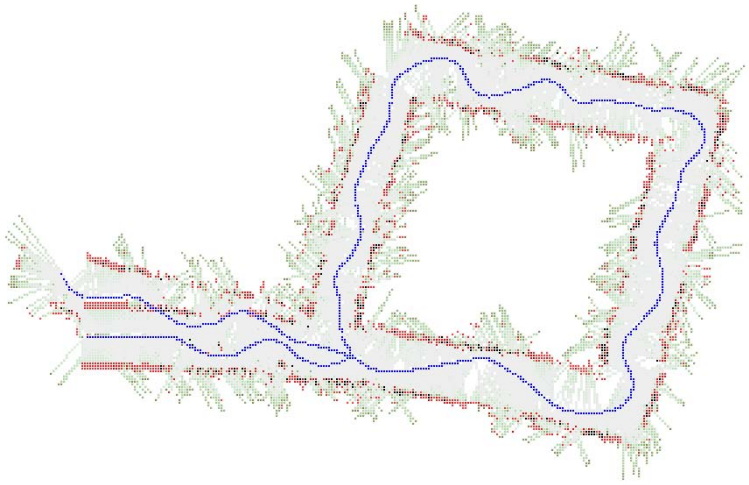


Figure 1.1: *Map demonstrating localization error*

information may then either be communicated back to intelligence sources or utilized for the delivery of certain payloads.

1.2 Problem Statement

The field of robotic mapping has matured to a point where detailed maps of complex environments can be built in real-time, specifically indoors. Many existing techniques are robust to noise and can cope with a large range of environments. However, the ultimate goal of mapping is to solve the *real-time* localization, or correspondence, problem, which is quite challenging and effectively unsolved even for static environments. The goal of this work is to tackle this very problem—to implement a robot mapping technique that uses a form of the Bayesian evidential method [33] for fusing sensor readings into a probabilistic representation of the environment, and to synthesize a variety of localization techniques that are currently used in the field.

1.3 Approach

The means by which we address this problem involves the building and representation of environments using a collection of independent Bayesian occupancy grids that together model

the environment using sonar range data. These individual grids, called local maps, cover a small area of the full environment, or global map, and are a key element to our localization approach. The expected pose (location and orientation relative to the global environment) of each map is calculated and maximized using a procedure that is akin to the alpha and beta functions of the Hidden Markov Model [36] in that they are computed separately, one forward as part of the mapping procedure and the second backward over those local maps previously acquired. Local maps are reconstructed using the maximized pose, and the global map automatically corrected as a result.

The theory upon which our work is based is foundational to off-line batch localization. With the focus of our research being real-time localization, we designed a localization algorithm that implements a scaled version of this theory, largely in the extent over which the beta step occurs. Rather than perform the backward calculation over all previous maps, we do so only over the previous one. Furthermore, our forward alpha calculation is one that is implicit in the recursive nature of the occupancy grid map, and not explicit.

1.4 Thesis Outline

In Chapter II, we present a brief history of robotic mapping and techniques used specifically for localization, as well as those that are most directly applicable to this research. We introduce three common approaches to mapping: metric, topological, and probabilistic, focusing in large part on the latter as the dominant technique used in the field. We discuss the mapping problem, and several challenges that stand in the way of solving it, such as sensor noise and correspondence. The chapter then focuses on specific probabilistic techniques of Kalman filters, expectation maximization, and hybrids that are used to solve different aspects of the mapping problem, presenting their basis, strengths, and weaknesses; and then presents some of the open problems that remain to be solved. Finally, we round out the introduction by looking at prior work in the areas of concurrent mapping

and localization and mapping with real-time localization, which serve as an underpinning for our research.

In Chapter III, we detail the theory behind our mapping and localization approach, along with our implementation details. In the context of mapping, we present the Murphy sonar model and the means by which raw sonar range readings from a Pioneer robot are mapped to a Bayesian occupancy grid, considering three different models: point of return, acoustic axis, and field of view. Turning from mapping, we next move into the theory behind our approach to localization, presenting the data, motion, and perception models that serve as our statistical basis. The layered map representation we utilize is next presented, with discussion of how it helps to achieve localization. Finally, we present the theory behind our expectation maximization approach as it applies to localization, and discuss our specific implementation of that theory.

In Chapter IV, we present the results of our implementation in the context of simulated and real world environments. We first look at mapping without localization, focusing on the many parameters that are built into our mapping system and their effects on our maps. Among these, we consider the three sonar models presented in Chapter III and those that specifically affect the Murphy model, such as how probabilities are modified in certain regions of the sonar field of view, and the disabling of certain sonars. In the second half of the chapter we look at the results of our localization implementation, discuss parameters that had the greatest impact on those results, and present several simulated and real-world illustrations.

Lastly, we discuss future work and extensions in Chapter V. As our localization results were not as successful as anticipated, we explore a number of means by which they may be improved, some simply and others more involved. We then offer our concluding remarks.

II. Related Work of Robotic Mapping and Localization

This chapter presents previous work related to robotic mapping and several key approaches to solving the mapping and localization problems [44] that are addressed in this research.

Research in robotic mapping has been an active pursuit for over two decades. The problem of robotic mapping is that of modeling physical environments using mobile robots and their sensors, generally considered one of the most important steps toward achieving truly autonomous robots. As of now, there are fairly robust methods for mapping environments, but these methods are all subject to a variety of non-real-world constraints, namely static, structured, and of limited size. When it comes to unstructured, dynamic, or large-scale environments, the problem is largely unsolved.

Most state-of-the-art mapping algorithms are probabilistic in nature. Some map in real-time [20]; others collect data first and then build the map off-line [27]. The best results tend to be achieved by those that are processed in multiple passes off-line. Some algorithms address the problem of correspondence between data points that are acquired at different points in time (that is, such as when the robot returns to an area of the environment it has already visited) [13,37], and others employ feature-detection [3,23], where objects in the environment must have a signature that makes them uniquely identifiable.

2.1 Historical Overview

In the 1980s and 1990s, mapping was largely divided into two approaches: metric and topological. Metric maps are most commonly represented as occupancy grids, introduced by Elfes and Moravec [19,20,32], which model the free and occupied space of the environment. The fine grained nature of these grids, used in many systems, provides a resolution that is helpful toward solving hard mapping problems (such as correspondence), although this comes at a computational

price. Topological mapping implements the notion of connectivity between significant places, such as rooms, where arcs from one place to the next are annotated with navigation information.

Since the 1990s, probabilistic techniques have dominated the area of robotic mapping that was largely initiated by Smith, et. al. [38,39], who introduced a statistical framework for simultaneously building a map and localizing such that the robot's pose is corrected in response to odometry errors. This procedure has come to be known as Simultaneous Localization and Mapping (SLAM) [15,18] and Concurrent Mapping and Localization (CML) [26,43]. One family of algorithms employing the probabilistic technique is that of Kalman filters, which are used to estimate the map and the robot pose [11,12,16]. These maps generally describe the location of landmarks, significant features in the environment, or large numbers of raw range measurements [27]. A second family of algorithms is based upon Dempster's expectation maximization algorithm [14,30]. These algorithms specifically target the correspondence problem, which is the problem of determining whether sensor measurements recorded at different points in time correspond to the same physical entity. A third family of probabilistic techniques look to identify objects in the environment, such as ceilings, walls, doors that might be open or closed, and furniture and other objects that move [4,28,29].

2.2 The Mapping Problem

The problem of robotic mapping is that of acquiring an accurate spatial model of an environment using sensors that enable a robot to perceive its environment. Sensors commonly used include cameras; sonar, laser, infrared, and radar technologies; tactile sensors, compasses, and GPS. All these sensors are subject to errors, or noise. The motion commands given during exploration serve as important information for building maps, since they convey information about the locations at which different sensor measurements were taken. Robot motion is also subject to errors, and the controls alone are therefore insufficient to determine a robot's pose.

A key challenge in robotic mapping arises from the nature of the measurement noise. Modeling problems, such as robotic mapping, are usually relatively easy to solve if the noise in different measurements is statistically independent. If this were the case, a robot could simply take more and more measurements to cancel out the effects of the noise. Unfortunately, in robotic mapping, the measurement errors are statistically dependent. This is in part due to the fact that reflectance in the environment will remain consistent, so that repeated readings at one location will not change the inherent noise in the reading. Also, errors in control accumulate over time, which affects the way future sensor measurements are interpreted. This latter factor is illustrated in Figure 2.1, which shows how a small rotational error on one end of a long corridor can lead to many meters of error on the other. As a result, whatever a robot infers about its environment is plagued by systematic, correlated errors. Accommodating such systematic errors is key to building maps successfully, and it is also a key complicating factor in robotic mapping. Many existing mapping algorithms are therefore complex, both mathematically and in their implementation.

The second complicating aspect of the robot mapping problem arises from the high dimensionality of the environments being mapped. For example, consider how many numbers it takes to describe an environment like a person's home. If confined to the description of major topological entities, such as hallways, intersections, rooms, and doors, a few dozen numbers might be

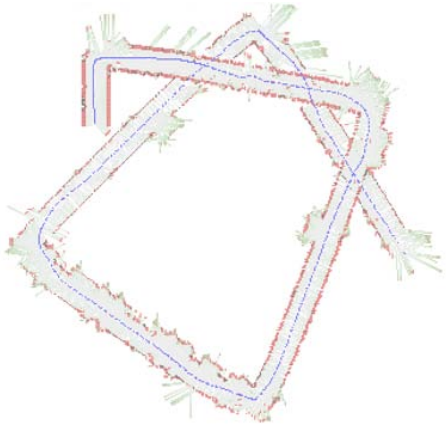


Figure 2.1: *Odometry error and the correspondence problem*

sufficient. A detailed two-dimensional floor plan, which is a common representation of robotic maps, often requires thousands of numbers. But a detailed 3D visual map of a building may easily require millions of numbers. From a statistical point of view, each such number is a dimension of the underlying estimation problem; thus, the mapping problem can be highly dimensional.

A third, and possibly the hardest problem, in robotic mapping is the correspondence problem, also known as the data association problem, which is the focus of this research. The correspondence problem is that of determining if an object or portion of the environment sensed at one point in time corresponds to the same object or environment sensed later in time. Figure 2.1 illustrates this problem as well, in which a robot attempts to map a large cyclic environment. When closing the cycle, the robot has to find out where it is relative to the map it has built so far. This problem is complicated by the fact that at the point of a cycle's closing, the robot's accumulated pose error might be unboundedly large. The correspondence problem is difficult, since the number of possible hypotheses can grow exponentially over time.

Fourth, environments change over time. Some changes may be relatively slow, such as the change of appearance of a tree across different seasons, or the structural changes that most office buildings are subjected to over time. Others are faster, such as the change of door status or the location of furniture items. Even faster may be the change of the location of other agents in the environment, such as cars or people. These dynamic environments create a big challenge, since they add yet another way in which seemingly inconsistent sensor measurements can be explained. For example, imagine a robot facing a closed door that previously was modeled as open. Such an observation may be explained by two hypotheses, namely that the door status changed, or that there is no door and the robot is not where it believes to be. Unfortunately, there are almost no mapping algorithms that can learn meaningful maps of dynamic environments. Instead, the predominant paradigm relies on a static world assumption, as we do here, in which the robot is the only time-variant quantity (and everything else that moves is just noise). Consequently,

most techniques are only applied in relatively short time windows, during which the respective environments are static.

A fifth and final challenge arises when robots must choose their way during mapping, commonly called robot exploration. Any viable exploration strategy has to be able to accommodate contingencies and surprises that might arise during map acquisition. For this reason, exploration is a challenging planning problem, which is often solved sub-optimally via simple heuristics. When choosing where to move, various quantities have to be traded off: the expected gain in map information, the time and energy it takes to gain this information, the possible loss of pose information along the way, and so on. Furthermore, and most importantly for our research, the underlying map estimation technique must be able to generate maps in real-time—an important restriction that rules out many existing approaches.

As noted above, the literature refers to the mapping problem often in conjunction with the localization problem, which is the problem of determining a robot's pose. The reason for suggesting that both problems (the problem of estimating where things are in the environment and the problem of determining where a robot is) have to be solved in conjunction has to do with the fact that both the robot localization and the map are uncertain. By focusing on just one, the other introduces systematic noise. If the robot's pose was known all along, building a map would be quite simple. If we already had a map of the environment, there already exist efficient algorithms for determining the robot's pose at any point in time [5, 21]. In combination, however, the problem is much more difficult.

Today, nearly all algorithms for robotic mapping employ probabilistic models of the robot and its environment, and rely on probabilistic inference for turning sensor measurements into maps. The reason for the popularity of probabilistic techniques stems from the fact that robot mapping is characterized by uncertainty and sensor noise. Perceptual noise is complex and not trivial to model. Probabilistic algorithms approach the problem by explicitly modeling different

sources of noise and their effects on the measurements. In the evolution of mapping algorithms, probabilistic algorithms have emerged as the sole winner for this difficult problem [44].

2.2.1 Occupancy Grid Maps. The simplest mapping algorithm assumes mapping with known poses. One mapping algorithm developed by Elfes and Moravec in the mid-Eighties [20,32], known as occupancy grid maps, has been used by a number of autonomous robots, typically in combination with one of the algorithms discussed below, and is used in this research as well.

The central problem addressed by occupancy grid mapping and related algorithms is the problem of generating a consistent metric map from noisy or incomplete sensor data. Even if the robot poses are known, it is sometimes difficult to say whether a place in the environment is occupied or not, due to ambiguities in the sensor data. The best-explored applications of occupancy grid maps require robots with range sensors, such as sonar sensors or laser range finders. Both sensors are characterized by noise. Sonars, in addition, cover an entire cone in space, and from a single sonar measurement it is impossible to say where in the cone the sensed object is. Both sensors are also sensitive to the angle of an object surface relative to the sensor and the reflective properties of the surface (absorption and dispersion).

Occupancy grid maps address such problems by generating probabilistic maps. As the name suggests, occupancy grid maps are represented by grids, which are usually two-dimensional. The standard occupancy grid mapping algorithm is a version of Bayes filters [44]. The occupancy grid mapping algorithm is recursive, allowing for incrementally updating the individual grid cells as new sensor data arrives.

Finally, calculating occupancy grid maps requires two probability densities, one that represents the prior probability of occupancy and another, the inverse sonar model, that specifies the probability that a grid cell is occupied based on a single sensor measurement. Inverse models for range sensors usually attribute high probability of occupancy for grid cells that overlap with detected range, and low probability for grid cells in between this range and the sensor.

2.2.1.1 Histogramic In-Motion Mapping. Histogramic In-Motion Mapping (HIMM) is part of a real-time obstacle avoidance system developed at Carnegie Mellon University that not only produces maps, but also provides instantaneous environmental data for use by an integrated obstacle avoidance system. The method represents data in a two-dimensional histogram grid (Figure 2.2), based on the occupancy grid, where belief of occupancy is represented by discrete histogramic values rather than probabilities. The system takes a range reading from each sonar and maps that reading to the specific cell that corresponds to the measured distance along the acoustic axis. The value in this cell is incremented by 3, and the values of all the cells preceding it along the acoustic axis are decremented by 1; minimum and maximum values are capped at 0 and 15, respectively. The primary purpose for using this approach in lieu of a probabilistic model is speed: the probabilistic certainty grid updates all cells within the cone of the sonar, which involves considerably more calculation. [6]

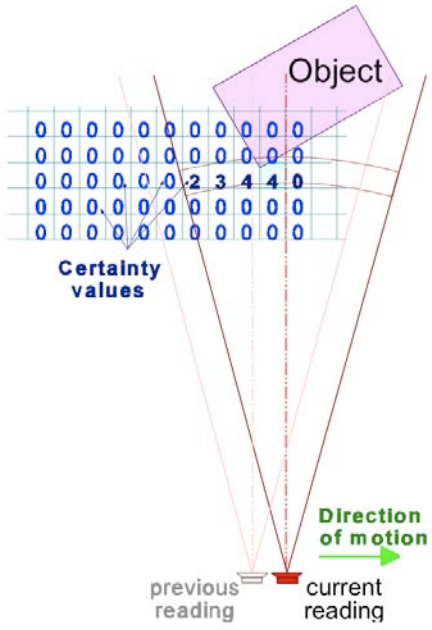


Figure 2.2: *Mapping an object using HIMM* [6]

2.2.2 Kalman Filters. A classical approach to generating maps is based on Kalman filters, referred to in the literature as Simultaneous Localization and Mapping [24]. This approach can

be traced back to a highly influential series of papers by Smith, Self, and Cheeseman [38, 39], who in 1985 through 1990 proposed a mathematical formulation of the approach that is still in widespread use today. In the following years, a number of researchers developed this approach further [11, 16, 18, 26].

Kalman filters are Bayesian filters that represent posteriors $P(s_t, m|z^t, u^t)$ with Gaussians [44], which models the probability of the robot's pose s at time t and the map m given the sensor reading z and robot control u, also at time t. In a two-dimensional case, the pose s is usually modeled by three variables: the Cartesian coordinates in the plane and the heading direction. The motion model (Section 3.2.1.2) is linear with added Gaussian noise, as is the perception model (Section 3.2.1.3). The map is represented as a collection of landmark locations, along with Gaussian uncertainty as to their location.

The primary advantage of the Kalman filter approach is that it estimates the full posterior[1] over maps in an online fashion, although pre-processing and filtering for landmark detection is necessary [44]. To date, the only algorithms that are capable of estimating the full posterior are based on Kalman filters or extensions thereof, such as mixture of Gaussian methods [18], or a Rao-Blackwellized particle filter [17, 31]. There are many advantages to estimating the full posterior. In addition to the most likely map and robot poses, Kalman filters maintain the full uncertainty in the map, which can be highly beneficial when using the map for navigation. Additionally, the approach can be shown to converge with probability one to the true map and robot position, up to a residual uncertainty distribution that largely stems from an initial random drift [16, 34]. As is commonly the case with theoretical results, they only hold under specific conditions and require that the robot encounter each landmark infinitely often. Even so, this is the strongest convergence result that presently exists in the context of robotic mapping.

[1]The full posterior probability is generally given as $P(H_i|E) = \frac{P(H_i)P(E|H_i)}{\sum_{k=1}^{m} P(H_k)P(E|H_k)}$, where the probability of a hypothesis given the evidence $P(H|E)$ is equal to the likelihood of the evidence given the hypothesis $P(E|H)$ multiplied by the prior probability of the hypothesis $P(H)$.

Probably the most important limitation of the Kalman filter approach lies in the Gaussian noise assumption, which states that the measurement noise must be independent and Gaussian. This has important implications for practical implementations. For example, in an environment with two indistinguishable landmarks, a multimodal distribution over possible robot poses will result, contrary to the unimodal Gaussian noise assumption. As a result, Kalman filter approaches are unable to handle the correspondence problem, a significant limitation for our research.

2.2.3 Expectation Maximization Algorithms. An alternative to the Kalman filter paradigm is known as the expectation maximization (EM) family of algorithms. EM is a statistical algorithm that was developed in the context of maximum likelihood estimation with latent variables in an influential paper by Dempster, Laird and Rubin [14].

Applied to the robotic mapping problem, EM algorithms constitute today's best solutions to the correspondence problem in mapping [9, 43]. In particular, EM algorithms have been found to generate consistent maps of large-scale cyclic environments, even if all features look alike and cannot be distinguished perceptually [44]. However, EM algorithms do not retain a full notion of uncertainty. Instead, they perform hill climbing in the space of all maps in an attempt to find the most likely map (not necessarily the true map). To do so, they have to process the data multiple times; therefore, EM algorithms cannot generate maps incrementally, as is the case for many Kalman filter approaches, and thus not in real-time, a requirement for this research.

For this approach, maps are represented as point obstacles, where points correspond to corners, intersections, and other distinctive places. These points are used to form a topological map representation, and overlayed with raw sensor data to build an occupancy grid map, illustrated in Figure 2.3. Existing implementations rely on grid representations for all densities involved in the estimation process, and also rely on probabilistic maps instead of zero-one maps (Section 2.2.1.1). The pose representation is the same as that for the Kalman filter approach; however, the sensor noise does not have the Gaussian restriction, but rather can be of any type.

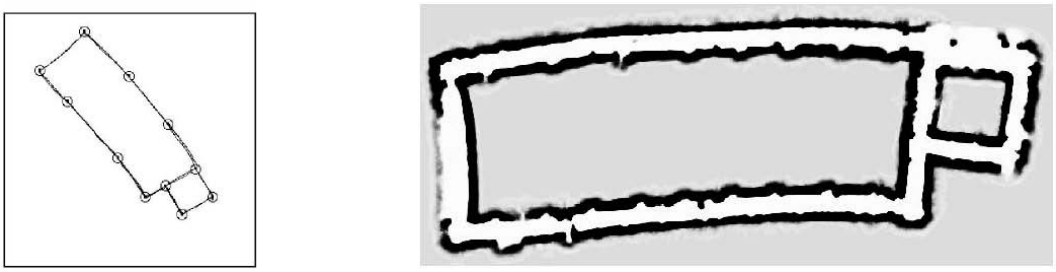


Figure 2.3: EM point obstacles [44]

The EM algorithm builds on the insight that building a map when the robot's path is known (in expectation) is relatively simple, as is finding a probabilistic estimate of the robot's location for a given known map. It does this by iterating over two steps: an expectation step (E-step), where the posterior over robot poses (the locations of possible robot poses) is calculated for a given map, and a maximization step (M-step), in which EM calculates the most likely map given these pose expectations, resulting in a series of increasingly accurate maps. The pose probabilities calculated in the E-step correspond to different hypotheses as to where the robot might have been, and suggest a number of possible correspondences with other mapped elements in the environment. By building maps in the M-step, these correspondences are translated into features in the map, such as walls or other obstacles, which then either get reinforced in the next E-step or gradually disappear.

Advantages of the EM algorithm over Kalman filtering include its ability to build maps from raw sensor data and most significantly, its ability to solve the correspondence problem (see Figure 2.4), although only in an off-line fashion. A scaled form of this procedure is used in this research.

2.2.4 Hybrid Approaches. There are many examples of hybrid solutions which integrate probabilistic posteriors with computationally more efficient maximum likelihood estimates.

One of the most common approaches is the incremental maximum likelihood method [20, 32,40]. The basic idea is to incrementally build a single map as the sensor data arrives, but without keeping track of any residual uncertainty. Such a methodology can be viewed as an M-step in

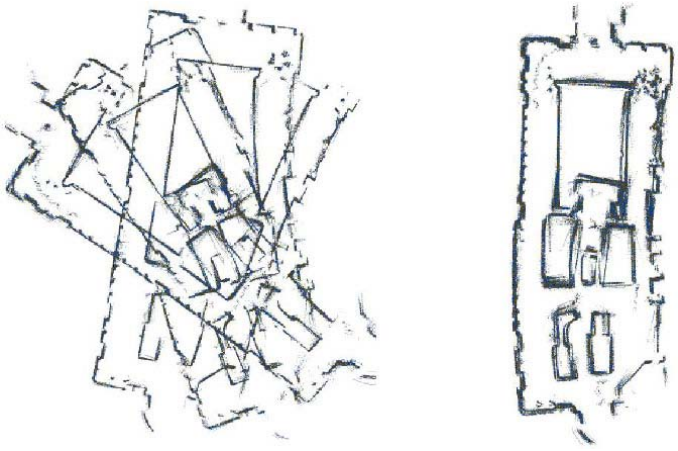


Figure 2.4: *EM applied to a cyclic environment*, showing raw range data of a large hall in a museum that is aligned using artificial but indistinguishable landmarks [44]

EM, without an E-step. The advantage of this approach lies in its simplicity, which accounts for its popularity.

Like Kalman filters, this approach can build maps in real-time, but without maintaining a notion of uncertainty. Like EM, it maximizes likelihood. However, the incremental one-step likelihood maximization differs from the likelihood maximization over an entire data set. In particular, once a pose and a map have been determined, they are frozen once and forever and cannot be revised based on future data—a key feature of both the Kalman filter and the EM approach.

This weakness manifests itself in the inability to map cyclic environments, where the error in the pose may grow without bounds. Figure 2.5 shows an example where the incremental maximum likelihood approach is used to map a cyclic environment, with a robot equipped with a laser range finder. While the map is reasonably consistent before closing the loop, the large residual error leads to inconsistencies that cannot be resolved by the incremental maximum likelihood approach. This is a general limitation of algorithms that do not consider uncertainty when building maps, and that possess no mechanism to use future data to adjust past decisions.

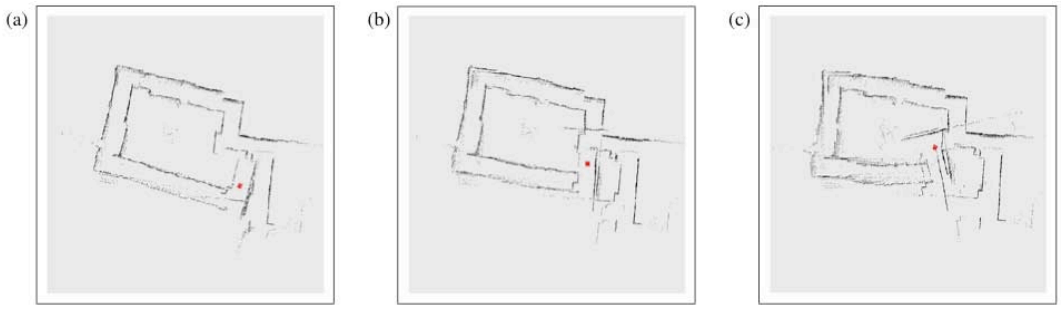


Figure 2.5: *Incremental maximum likelihood applied to map of cyclic environment* [44]

Hybrid approaches overcome this limitation by maintaining an explicit notion of uncertainty during mapping, short of the full posterior over maps and poses maintained by Kalman filters. A good example are the algorithms described in [22, 41, 42], which use the incremental maximum likelihood approach to build maps, but in addition maintain a posterior distribution over robot poses. The idea is that by retaining a notion of the robot's pose uncertainty, conflicts like the one faced by the incremental maximum likelihood method can be identified, and the appropriate corrective action can be taken. Figure 2.6 shows a sequence of map estimation steps using this approach using the same data as that for Figure 2.5. When the robot traverses a cyclic environment, it uses the samples to localize itself relative to the previously built map. When it has determined its pose with high likelihood, it uniformly spreads the resulting error along the cycle in the map. Consequently, the approach still maintains just a single map, which is computationally advantageous. But unlike the incremental maximum likelihood methods, it also has the ability

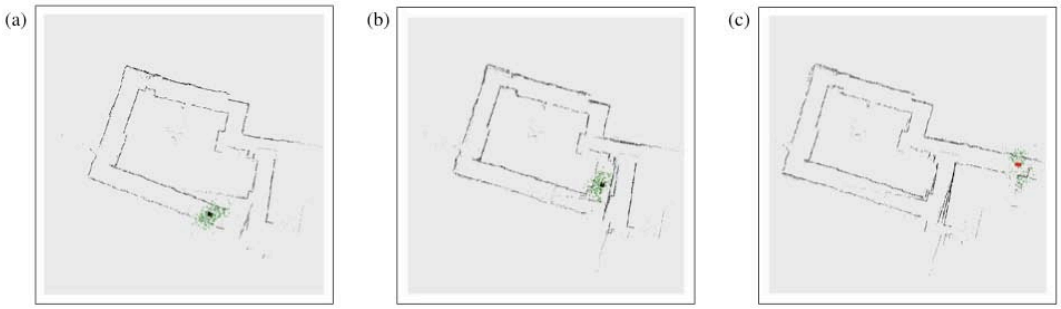


Figure 2.6: *Hybrid mapping applied to map of cyclic environment* [44]

to correct its map backwards in time whenever an inconsistency is detected. Mathematically, this hybrid algorithm can be derived as a rather crude approximation to the EM algorithm, which performs the E-step and the M-step selectively, as discrepancies are detected.

The hybrid approach suffers many deficiencies. First and foremost, the decision to change the map backwards in time is a discrete one which, if wrong, can lead to catastrophic failure. Moreover, the approach cannot cope with complex ambiguities, such as the uncertainty that arises when the robot traverses multiple nested cycles. Finally, the hybrid approach is—strictly speaking—not a real-time algorithm, since the time it takes to correct a loop depends on the size of the loop. However, practical implementations appear to work well in real-time when used in office-building type environments.

Since our implementation is a hybrid approach itself, it shares many of the characteristics described here, particularly in its incremental nature, its vulnerability to cycles, and its discrete decision to change a map backwards in time. However, it differs in that a local map's pose is not once determined and thereafter constant; rather, it may be updated as a result of future localizations. Also, rather than first collecting a map and then localizing off-line, our approach maps and localizes in real time.

2.2.5 Open Problems. After nearly three decades of research, the field of robotic mapping has matured to a point where detailed maps of complex environments can be built in real-time, specifically indoors. Many existing techniques are robust to noise and can cope with a large range of environments. Nevertheless, there still exist a large number of challenging open problems.

In particular, most published methods assume that the world does not change during mapping. Real environments are dynamic, such as would be the case for personal service robots. A second shortcoming of current technology arises from the vast amount of knowledge that we, as people, possess about environments, but that is presently not applied to the mapping problem, due to our inability to algorithmically process voluminous amounts of data.

From a statistical point of view, the ultimate goal of mapping is to solve the correspondence problem. While this problem is hard and effectively unsolved even for static environments, it is even more challenging for dynamic environments.

Another dimension worth exploring is the topic of unstructured environments. Most examples in this survey focused on indoor environments, which possess a lot of structure. Outdoor, underwater, and planetary environments also possess some structure, but come in a much larger variety. In the extreme, the environment may be entirely unstructured, such as piles of rubble that a robot might want to explore after a natural disaster or a terrorist attack. Many of the techniques described here are inapplicable.

2.3 Research-Specific Literature

Of the general research presented thus far, there are specific endeavors that serve as a strong foundation for this research, addressing a probabilistic approach to real-time concurrent mapping and localization.

2.3.1 Concurrent Mapping and Localization. Thrun, et. al. [43] address the problem of building large-scale geometric maps of indoor environments, posing the map building problem as a constrained, probabilistic maximum likelihood estimation problem. The general problem they address is how a robot can construct a consistent map of the environment if it only occasionally observes a landmark (as opposed to frequently), specifically when those landmarks may be indistinguishable and where the accumulated odometric error may be large. They present an algorithm for generating the most likely map from the data, along with the most likely path taken by the robot. Their results are effective in cyclic environments of up to 80 x 25 meters.

The paper presents an algorithm for landmark-based map acquisition and concurrent localization that is based on a statistical account of robot motion and perception. The problem of map building is posed as a maximum likelihood estimation problem, where both the location of

landmarks and the robot's position have to be estimated. Likelihood is maximized under probabilistic constraints that arise from the physics of robot motion and perception. The problem is solved efficiently using the Baum-Welch (or alpha-beta) algorithm [35], which is a special version of EM [14] in how it alternates the E-step and M-step (sometimes also called modification step). In the E-step, the current map is held constant and the probability distributions are calculated for past and current robot locations. In the M-step, the most likely map is computed based on the estimation result of the E-step. By alternating both steps, the robot simultaneously improves its localization and its map, which leads to a local maximum in likelihood space. The probabilistic nature of the estimation algorithm makes it considerably robust to ambiguities and noise, both in the odometry and in perception. It also enables the robot to revise past location estimates as new sensor data arrives.

While the theory presented in this work, detailed in Section 3.2.1, is a key component of the research here, the model does operate under several assumptions or restrictions. One, it assumes that a robot operator selects a small number of significant places (landmarks such as intersections, corners, or dead ends), where the robot is informed each time such a place has been reached; that the robot can observe landmarks; and that it has the means for estimating their type, relative angle, and approximate distance. Also, in their model of robot perception (the probability of an observation given the current pose and map of the environment), the map contains knowledge about the exact location of all landmarks, leaving the pose alone unknown. Furthermore, the localization is not conducted simultaneously with the mapping in real time.

In follow-on work, Burgard, et. al. [10], extended this research by changing their single-map into a layered representation of maps, where a global map is composed of a collection of small, local maps that are generated from short sequences of sonar data . This approach is based on the notion that short-term odometry errors are typically small and safely ignored. These local maps, annotated with their pose distributions, take the place of the landmarks used in the previous

approach. This approach allows for the maintaining of dependencies between different grid cells within each local map, which is in contrast to the independence assumption inherent in the single global map. The result is they are able to use conventional metric Markov localization in the E-step, and are able to simultaneously operate on raw proximity information without relying on pre-defined landmarks. Their empirical results illustrate success in learning large-scale maps based solely on sonar range data.

While it is this notion of local maps that we implement within this research, and much of the theory upon which their EM is based, their algorithm does remain an off-line batch localization. As a result, our research implements a scaled version so as to achieve real-time localization.

In additional CML work to which our research is directly related, Laviers [25] uses an occupancy grid to model individual rooms while they are being mapped, and then upon completion of each room, converts them into a polygonal representation. These reduced-information environment representations are further reduced to a connected graph of Absolute Space Representations (ASRs). The approach performs real-time localization on the ASRs using EM, localizing the individual rooms to each other; there is no localization occurring within the rooms. One goal of our research is to provide this in-room localization.

2.3.2 Mapping with Real-time Localization. The mapping and localization techniques presented in [45] are specifically relevant to our research for the means by which they build the occupancy grid, determine a robot's expected pose, and use the notion of local maps to build a single global map.

While the localization method requires as input a global map that identifies occupied cells, it is their approach to determining where the robot is in the global map that has application here. In brief, a single sonar range sweep is taken, itself considered a single local map, and the range vectors generated by the sweep are compared against all locations on the global map. Those that are feasible poses, that is, those where the vector is not blocked by an obstacle are added to a

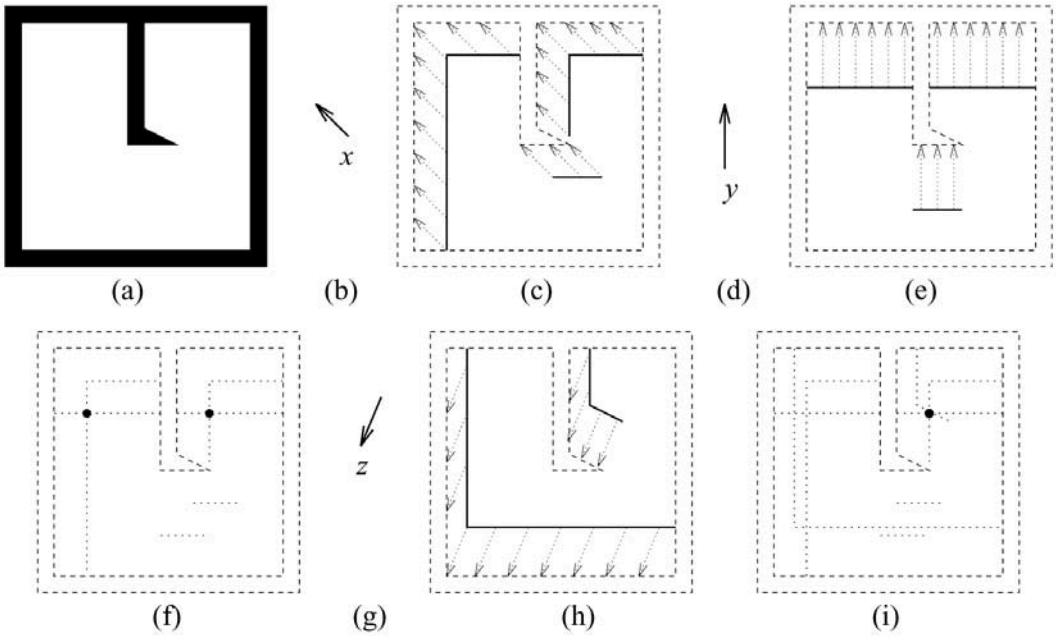


Figure 2.7: *Localization using feasible poses over a known map* [8]

collection of feasible poses. The intersection of these poses identifies the location of the robot. This is illustrated in Figure 2.7: Given a map M (a) and a range vector x (b), they determine the set of feasible poses $FP(M, x)$ (c); along with range vector y (d) and feasible poses $FP(M, y)$ (e), the intersection of poses $FP(m, \{x, y\})$ (f) narrows those feasible down to two; and finally, range vector z (g) is added with feasible poses $FP(m, z)$ (h), to yield the singular feasible pose (i). [8]

The results presented by Varveropoulos are impressive, but a key shortcoming is that preprocessing using a global map is required [2]. And although [8] address localization without explicit landmarks, they too are using a global map and a robot that is manually positioned at various locations within the environment.

2.4 Contribution of this Study

With the significant progress that has been made in autonomous robotic mapping, each approach makes several assumptions or is subject to various constraints so as to break the larger problem down into smaller bits that may one day be recombined into a far-distant mapping

algorithm that makes no assumptions and has no constraints. For this research, our goal is to map a static environment using sonar with real-time localization in a local context, that is, in the context of a single room, that is to be combined with prior work in global localization using absolute space representations (ASR).

III. Methodology

In this chapter we present the theory and implementation central to our research. Our mapping algorithm is an implementation of a Bayesian occupancy grid that uses a linear sonar model. We generate three different representations of this model that assist in producing maps most conducive to localization. Our localization is based upon an HMM-based EM approach that utilizes local maps to correct for robot pose error.

3.1 Mapping

The occupancy grid discussed in section 2.2.1 is one component of our implementation, which is based on the work of Murphy [33, pp 380-386]. In this model, an ultrasonic range reading is mapped to a specific cell on the occupancy grid, and probabilities of occupancy are calculated for that cell and those in surroundings regions.

3.1.1 The Sonar Model. Fundamental to this research is an understanding of the sonar sensor model, illustrated in Figure 3.1, and detailed by [33, pp 378-380]. A wave emits from the sonar device along the acoustic axis that spreads and generates a field of view of approximately 30°, as determined by the Polaroid sonars on the Pioneer hardware we utilize. Within a certain maximum range, the wave will bounce off an object and return to the sonar sensor, at which point a distance to that object will be calculated based on that wave's time of travel. This distance is measured along the acoustic axis, the line bisecting the cone; the object's angle off the acoustic axis is not measurable.

This beam can be projected onto a grid, as shown in Figure 3.1. This is commonly called an occupancy grid because a value can be calculated for each cell, representing the likelihood that the cell is occupied.

While the sonar reading is identifying an object at the intersection of the acoustic axis and the center of the darkened region in the figure, call it C_c, the noisy nature of the sonar leads to

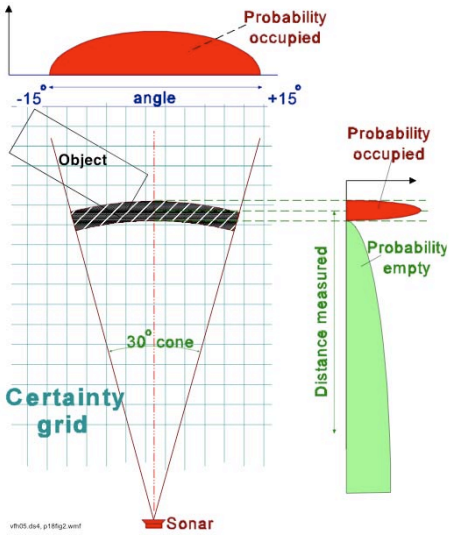


Figure 3.1: *The probabilistic sonar model over an occupancy grid* [7]

the possibility that the object is actually off the axis or slightly behind or in front of the region's center. In a probabilistic occupancy grid, this is acknowledged by calculating an increasingly lower probability of occupancy for the area surrounding C_c, represented in Figure 3.1(a) by the darkly shaded (red) domes. As shown, the distribution is ideally Gaussian. For efficiency, however, Murphy employs a linear distribution, which we implement here (Figure 3.2).

Murphy details the sonar model by identifying some key nomenclature representing various measures and regions. The half-angle of the sonar beam is specified by β, and the maximum range of the sonar is specified by R. Additionally, a particular sonar beam is divided up into three regions:

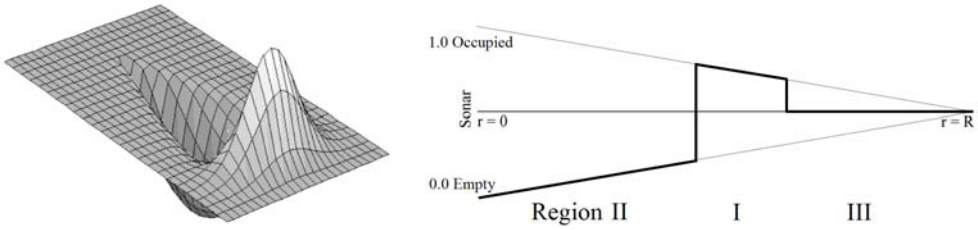
Region I: *the narrow area around the sensed object, for which there is a high confidence of occupancy*

Region II: *the area preceding Region I, for which there is a high confidence of vacancy*

Region III: *the area beyond Region I, for which no estimation of occupancy is made*

When considering the cells that fall within either Region I or II, another key component is the notion of a cell's distance off the acoustic axis, specified by α, and its distance from the sensor, specified by r.

With these four terms, and knowledge of which region a cell falls into, a probability that a particular cell is occupied given a sonar reading is calculated. This probability model is illustrated in Figure 3.2(b), where the closer to the sonar device a cell is, the higher the probability it is either occupied (Region I) or empty (Region II).



(a) (b)

Figure 3.2: *Modeling sonar probabilities*, showing full Gaussian (a) and linear (b) representations.

The first step is to calculate the probability that the sonar would be returning the reading given the cell is actually occupied, $P(s|H)$, where H is the hypothesis that the cell is occupied. For cells that fall within Region I, that is, in the region of the arc in the immediate vicinity of C_c, this probability is given as:

$$P(s|H) = \frac{\left(\frac{R-r}{R}\right) + \left(\frac{\beta-\alpha}{\beta}\right)}{2} \times Max_{occupied} \tag{3.1}$$

where the constant $Max_{occupied}$ represents a cap on the certainty that can be assigned to any cell, such as 0.98 (the value used for this research). For cells that fall within Region II, that is, the region preceding Region I, we are reasonably certain they are empty, else we would not have received a range reading beyond them. Thus the probability that they are occupied is given as 1 minus the probability of being occupied:

$$P(s|H) = 1 - \frac{\left(\frac{R-r}{R}\right) + \left(\frac{\beta-\alpha}{\beta}\right)}{2} \tag{3.2}$$

While in the Murphy model the absence of $Max_{occupied}$ reflects there is no need to cap the probability that a cell is empty, in this implementation, we do experiment with a cap, namely 0.5, which captures the notion that cells in Region II are assumed to be empty, thus their probability of occupancy should not exceed 0.5. Without this cap, probabilities of occupied within this region are allowed to go to 1.0.

Because the probability that we are actually interested in is that of a particular cell being occupied given a sonar reading, and not the reverse, Bayes' rule must be applied:

$$P(H|s) = \frac{P(s|H)P(H)}{P(s|H)P(H) + P(s|\neg H)P(\neg H)} \tag{3.3}$$

where $P(H)$ and $P(\neg H)$ are the prior probabilities that the cell is occupied and empty, respectively, and $P(s|\neg H) = 1 - P(s|H)$. The actual value of $P(s|H)$ will vary depending upon which region the cell maps to, as discussed above.

As concerns the priors, before the environment has been explored, an assumption is made that there is an equal probability that any cell is either occupied or empty; that is, $P(H) = P(\neg H) = 0.5$, and the entire occupancy grid is initialized to 0.5. Thus the prior is the preexisting value of the cell under consideration, and the resulting $P(H|s)$ becomes the new value (prior) stored in that cell.

3.1.2 Mapping Range Data to the Occupancy Grid. Mapping is performed using the Pioneer P2-AT robot, which has a particular sonar configuration and pose representation that do not correspond directly to the occupancy grid domain.

First among these is the manner in which sonars are oriented on the robot (reference Figure 3.3). The sonars are numbered clockwise from 0 to 15, with 0 located at the 9 o'clock position, or the western-most position relative to the robot's heading. Second, the heading of the robot is reported as being between 0 and ±179 (interior numbers in the figure), with a positive value indicating the robot's right side, and a negative indicating the left side. Third, the unit of measure used for range

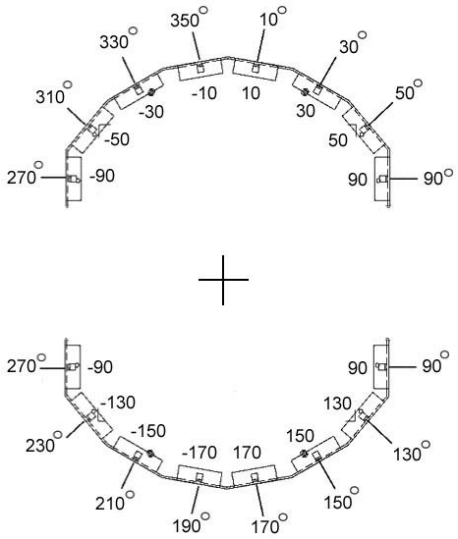


Figure 3.3: *Pioneer sonar array*

readings is given in millimeters. Lastly, the position and heading of the robot, $(x, y) : \theta$, at the time a run begins are $(0, 0) : 0^o$, regardless of what the robot's actual orientation is in a simulated world or a real-world lab.

Another implication of this unknown initial position is that the data structure used to store the grid world must take into account that the robot may start at any extreme within its environment. For example, if its initial position were the bottom-left-most corner of a 50 x 100 world (5 x 10 meters), the world would be bounded by (0,0) and (50,100); whereas an initial top-right-most position would bound the world by (-50,-100) and (0,0). Since the initial orientation is unknown, if a static data structure is used, it must be four times as large as the actual world, allowing for unhindered travel in any direction, in this case, (-50,-100) and (50,100). In order to reduce this typically heavy memory requirement for occupancy grid mapping, we chose to implement a dynamic structure. Our mutable Cartesian grid starts out with no dimension, and as it is accessed, it automatically grows to accommodate those coordinates that are outside its current bound.

While the Pioneer robot returns several range sweeps per second, we are recording only one per distance traveled within a grid cell, which is 100mm square, or about one reading every 4

inches. The reasoning behind this decision lies in the consistency of the sonar noise. As mentioned in Section 2.2, the reflectance of the sonars remains consistent throughout the environment. Furthermore, using this approach simplifies the model in that only one sweep need be mapped to a single cell in the grid world. We do consider in Section 5.1, however, extending our implementation to integrate multiple sweeps into a single range reading.

With this framework in place, there comes the task of mapping a given sonar reading to the grid world data structure. Given is the robot's pose, ξ, which as discussed, consists of it's x-y coordinate in the physical or simulated world and it's heading, θ, relative to its initial heading, as well as the number of the sonar returning the reading. Each sonar emits its signal at a known angle. Thus, using the robot heading and sonar heading, the position of the object can be determined using the simple coordinate transform:

$$x_s = x_r + d_s \sin(\theta_r + \theta_{rs}) \tag{3.4}$$

$$y_s = y_r + d_s \cos(\theta_r + \theta_{rs}) \tag{3.5}$$

$$x_o = x_s + d_o \sin(\theta_r + \theta_s) \tag{3.6}$$

$$y_o = y_s + d_o \cos(\theta_r + \theta_s) \tag{3.7}$$

where (x_r, y_r), (x_s, y_s), and (x_o, y_o) identify the coordinates of the robot, sonar device, and object, respectively; and θ_r, θ_{rs}, and θ_s identify the heading of the robot, the angle from the robot center to the sonar device, and the heading of the sonar signal relative to the robot, respectively. It is this grid world coordinate (x_o, y_o) that corresponds to the object and, along with the grid world coordinate of the robot's center point, serves as the frame of reference for updating the occupancy grid with the result of Equation (3.3).

3.1.3 Building the Occupancy Grid. There are three increasingly complex approaches, or models, for updating the occupancy grid that are implemented in this research:

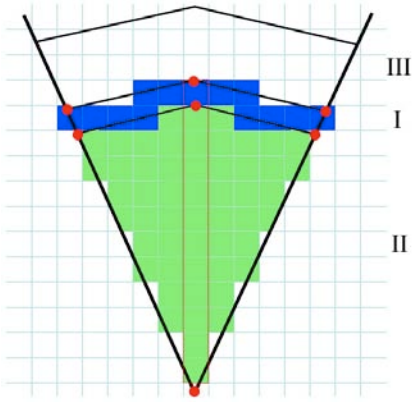


Figure 3.4: *Mapping the sonar probability model using polygon filling*

Point of Return Update the cell corresponding to the range reading, C_c

Acoustic Axis Update all the cells that fall along the sonar's acoustic axis

Field of View Update all the cells within the cone of the sonar beam

The first approach has the benefit of being extremely fast, quite simple, and rather effective, at least for generating a good estimate of the environment (discussed in Chapter 4). It is a version of the HIMM approach discussed in Section 2.2.1.1, with the difference that it uses probability values rather than histograms.

The second approach employs the Bresenham Line Algorithm [46] to traverse the cells along the acoustic axis, that is, between (x_o, y_o) and (x_r, y_r) (Figure 3.4). As each cell is traversed, the cell is filled with its probability of occupancy, taking into account its distance r from the sensor (the angle α off the acoustic axis is always 0 for this method).

The third approach, also illustrated in Figure 3.4, represents a full implementation of the probabilistic occupancy grid in that it prescribes that all cells within the sonar cone have probabilities associated with them for a particular reading. For this problem, a polygon filling routine is implemented, where two polygons are defined around Regions I and II, as defined by the points shown in Figure 3.4.

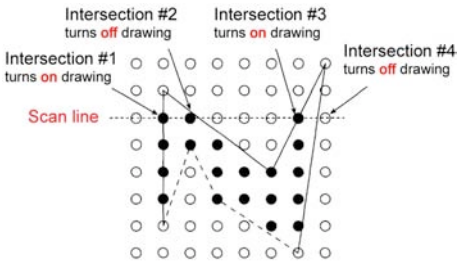


Figure 3.5: *Scan line polygon fill*

There are three considerations that went into choosing an appropriate polygon filling algorithm, namely; that it be efficient, complete, and non-redundant. Complete means that all cells within the polygon and between adjacent polygons must be filled and there may be no gaps. Non-redundant means no cell be filled more than once (even those that share a common border between two polygons), for in our recursive probabilistic occupancy grid context, a multiply-updated cell will result in an artificially high or low occupancy value. An algorithm that fulfills all three requirements is the Abrash implementation of the Scan Line Algorithm [1]. As shown in Figure 3.5, the concept entails vertically moving a horizontal scan line over the polygon in one direction, filling the cells that fall within the boundaries of the polygon lines it intersects, excluding those cells that fall on the off borders.

3.2 Localization

Localization describes the procedure by which corrections are made to the robot's pose such that its perceived location matches as closely as possible with its actual position. The further a robot travels, the greater the error in its expected pose. This error is most apparent in cyclical environments, such as when the robot maps a room beginning and ending at the same point, or travels through a corridor that circles back to the starting position. The latter is most difficult because in this situation, there is no prior knowledge in the map—no overlap or backtracking, so to speak, that assists in determining an accurate pose.

The localization approach used for this research is largely based upon the models presented in [10, 43, 45].

3.2.1 Statistical Basis. The problem of concurrent mapping and localization is here represented as a maximum likelihood estimation problem, where a number of expected poses for each local map given the data in the global map are found, and the maximum of those is selected.

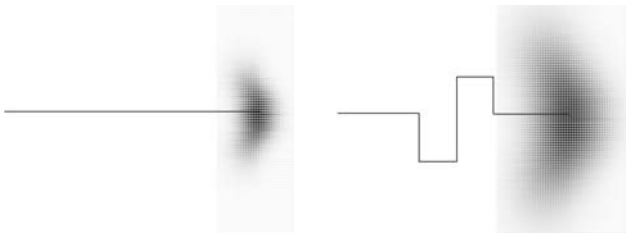
3.2.1.1 The Data Model. Following the nomenclature of [10], the stream of sensor data received from the robot is represented by

$$d = \{o^1, u^1, o^2, u^2, ..., o^T, u^T\}, \tag{3.8}$$

where o^t and u^t are a sweep of sonar observations and robot control commands at time t, respectively, the total number of time steps being represented by T. For this implementation, a control command at time t is equivalent to the robot's pose at that time (reported by the robot's internal position-speed encoders), because the Pioneer does not take discrete commands such as, "turn left 10 deg and move forward 100cm," but is rather controlled by a keyboard or joystick interface.

A map, denoted m, is an assignment of properties to x-y-locations in the world. While for [43], m contains knowledge of the exact location of all landmarks, for [10] these landmarks are replaced by local maps, which is the implementation used here. Furthermore, for the metric occupancy grid approach used here, these properties are probabilities that collectively represent obstacles.

3.2.1.2 The Motion Model. The motion model, denoted $P(\xi'|u, \xi)$, describes the probability of the current pose ξ' given the prior pose ξ and motion control command u. In the context of local maps, the prior pose is that of the prior local map, and the motion control command is a representation of the travel conducted over the course of that map. This model captures the



(a) (b)

Figure 3.6: *The motion model*, showing the further the distance of robot travel, the greater the pose distribution's depth; and the greater the degrees of turn made by the robot, the greater the distribution's width.

normally distributed uncertainty in the robot's known pose that increases the further the robot travels and the greater the degree of turns it makes, as illustrated by Figure 3.6. The shaded area shows the pose distribution (projected into two dimensions) after the path indicated by the solid line has been traveled, the degrees of gray reflecting the pose likelihood.

In this implementation, four factors go in to the construction of this model for any given local map pose:

- height, which reflects the robot's distance of travel in the local map
- width, which reflects the robot's degree of turn over the course of the local map
- σ, a Gaussian scale parameter for which a higher value stretches the distribution (values in the periphery are higher; the descent is slower; $\sigma = 1$ for a standard normal distribution)
- τ, which specifies the intensity of the model's triangular shape

Two Gaussian distributions corresponding to the width and height are used to form the distribution; σ is applied to both. Without τ, the resulting distribution would be elliptical; with τ, the sides of the distribution are effectively pulled down to form a triangular shape. Picturing the distribution as a matrix, this is accomplished by shifting entire columns down by a factor of the column's distance from the distribution's center and of τ. Thus, those columns on the outer edges are shifted by the greatest amount; those toward the center are shifted only slightly. A greater τ

produces a greater shift. While the width and height are factors of the robot's travel, σ and τ are static, yet configurable, system parameters.

3.2.1.3 The Perception Model. This model, denoted as $P(o|m, \xi)$, models the likelihood of observing o in situations where both the world map m and robot pose ξ are known. Here, this uncertainty in the sonar reading is being captured by the Murphy model previously described, which is largely a factor of range reading distance, and less so of a cell's distance off the acoustic axis, when considering the full sonar cone.

These three quantities: the set d of data, the motion model $P(\xi'|u, \xi)$, and the perception model $P(o|m, \xi)$, form the statistical basis for our localization approach.

3.2.1.4 The Map Likelihood Function. The problem of mapping can be stated as the problem of finding the most likely map given the data, that is, $m^* = \underset{m}{\text{argmax}}\, P(m|d)$. This probability can be derived as follows into the separate motion and perception models used for our localization approach:

$$m^* = \underset{m}{\text{argmax}}\, P(m|d) \tag{3.9}$$

$$P(m|d) = \int \dots \int P(m|\xi^1, \dots, \xi^T, d)\; P(\xi^1, \dots, \xi^T|d)\; d\xi^1 \dots d\xi^T \tag{3.10}$$

$$P(m|\xi^1, \dots, \xi^T, d) = \frac{P(d|m, \xi^1, \dots, \xi^T)\; P(m|\xi^1, \dots, \xi^T)}{P(d|\xi^1, \dots, \xi^T)} \tag{3.11}$$

$$P(d|m, \xi^1, \dots, \xi^T) = \prod_{t=1}^{T} P(o^t|m, \xi^t) \tag{3.12}$$

$$P(\xi^1, \dots \xi^T|d) = \prod_{t=1}^{T-1} P(\xi^{t+1}|u^t, \xi^t) \tag{3.13}$$

$$P(m|d) = \int \dots \int \frac{1}{P(d|\xi^1, \dots, \xi^T)} \prod_{t=1}^{T} P(o^t|m, \xi^t)\; P(m) \prod_{t=1}^{T-1} P(\xi^{t+1}|u^t, \xi^t)\; d\xi^1 \dots d\xi^T \tag{3.14}$$

$$= \underset{m}{\text{argmax}} \int \dots \int \prod_{t=1}^{T} P(o^t|m, \xi^t) \prod_{t=1}^{T-1} P(\xi^{t+1}|u^t, \xi^t)\; d\xi^1 \dots d\xi^T \tag{3.15}$$

where,

- $P(m|d)$ in (3.9) can be written as (3.10) by explicitly specifying and integrating over the pose information, recognizing that the map's likelihood is dependent upon those poses, and by applying Bayes' rule
- the first term on the right hand side of (3.10) can be rewritten as (3.11) via another application of Bayes' rule
- the first term on the right hand side of (3.11) can be transformed to (3.12), based on the observation that o^t depends only on the map m and the location ξ^t at time t, and not on prior observations
- the second term on the right hand side of (3.11) may be considered as $P(m)$ since in the absence of any data, m does not depend on any of the locations ξ^t, where $P(m)$ is the Bayesian prior over all maps, which is assumed to be uniformly distributed
- the second term in (3.10) can be rewritten as (3.13), based on the observation that the robot's location ξ^{t+1} depends only on the robot's location ξ^t one time step earlier and the action u^t executed at that point
- substituting (3.11), (3.12), and (3.13) into the right hand side of (3.10) leads to the likelihood function (3.14), and
- (3.15) results from substituting (3.14) back into (3.9) while dropping $P(m)$ and $1/P(d|\xi^1, ..., \xi^T)$, since we are only interested in maximizing $P(m|d)$ and not in computing its value

The final equation (3.15) is exclusively a function of the data d, the perceptual model $P(o|m, \xi)$, and the motion model $P(\xi'|u, \xi)$, all previously described. Maximizing this expression is equivalent to finding the most likely map. [43]

3.2.2 Layered Map Representation. To assist with real-time localization, occupancy grid values are mapped to a series of local maps which overlay atop a single global map, according to the theory of [10], which argues that the independence assumption inherent in occupancy grids

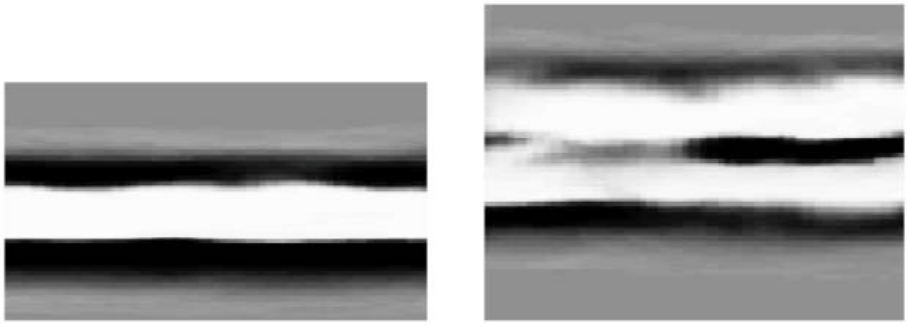


Figure 3.7: *Bi-modal pose distribution*

can be avoided by adopting a richer, layered representation using local maps. The problem they describe is illustrated in Figure 3.7: Suppose a series of robot sensor readings suggest a map like that shown on the left side, but it has two distinct hypothesis as to where it might be. Integrating this local map into a single global map in relation to those two hypothesis may yield a map like that shown on the right. Such maps are not usable for localization with proximity sensors, and the situation worsens when the robot is more globally uncertain, resulting in maps that often contain no visible structure.

The local maps address this problem. Each map represents the range data received over a portion of the global map. Each map maintains a pose that is taken as the robot pose in the global map at the time of creation, retains all the sonar range readings with which it is built, and tracks the distance and degrees of turn the robot traveled within its bounds. While the path of the robot within each local map is unique to that map (it is not repeated in any other), there is overlap between them due to the reach of the sonars. At construction, a local map is localized based upon the global map as it exists at that point. It is then built using sonar data for a prescribed distance and relocalized using the newly acquired data. This process repeats for the duration of the mapping session.

Rather than maintain a global map that convolves the local maps into a single monolithic map as they are created, the local maps are dynamically convolved whenever a cell in the global map is accessed. For example, if two maps overlap at cell (x, y), both maps are consulted for their

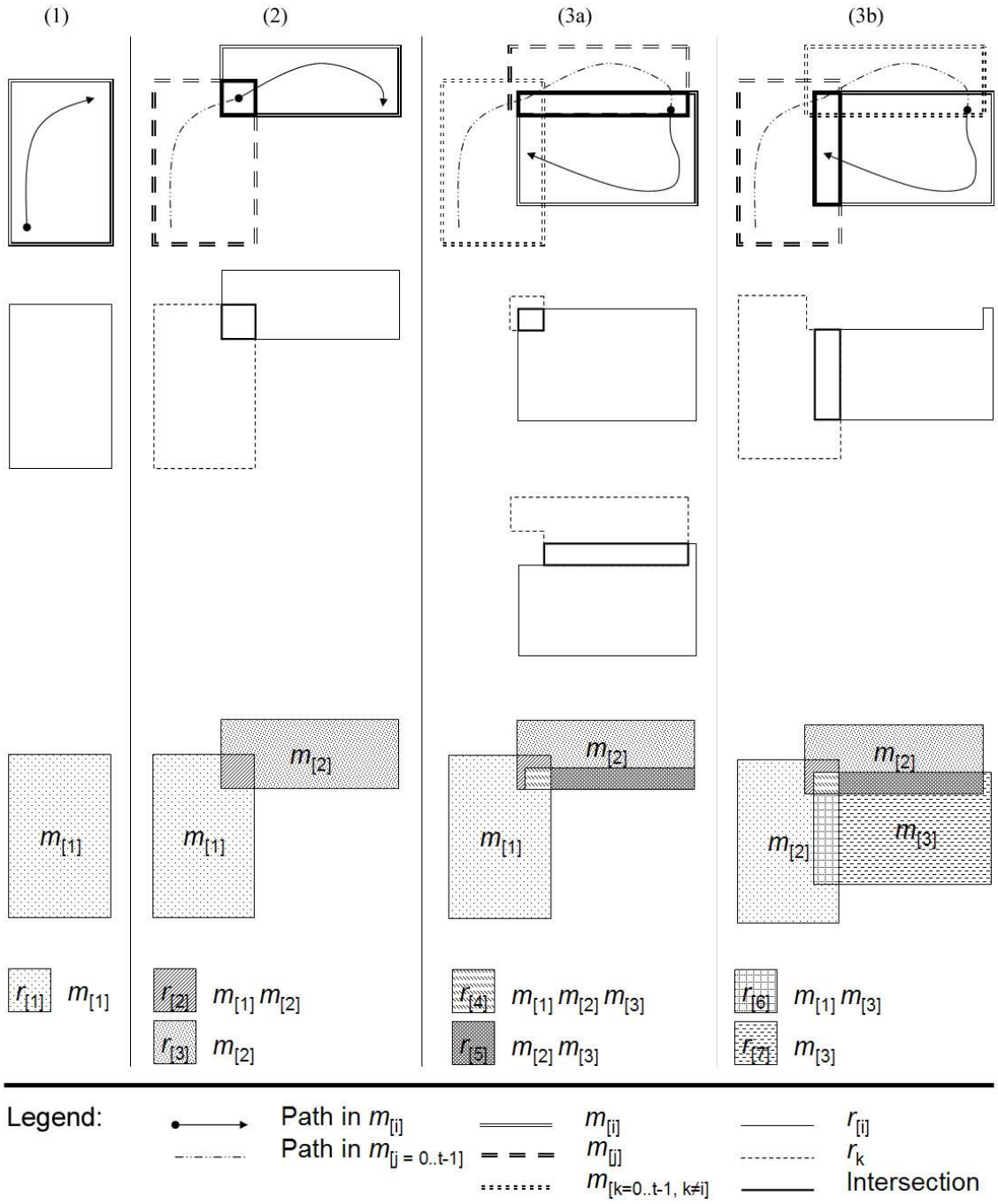


Figure 3.8: *Overlapping local maps regions*, showing construction of regions and mapping global coordinates to those local maps that cover the coordinates.

Algorithm 1 *Local map construction*

1: **procedure** MAPTOREGIONS($m_{[i]} : m, [R], r$)
2: $b_i \leftarrow m_{[i]}$.bound
3: $h \ll r_{[0]}$
4: **for all** $m_{[j],j=i-1..0}$ **do**
5: $b_{ij} \leftarrow b_i \cap m_{[j]}$.bound
6: **if** $b_{ij} \neq \emptyset$ **then**
7: **for all** cells $k \in b_{ij}$ **do**
8: $r_k \leftarrow [R]_k$
9: **if** $r_k \notin h$ **then**
10: $b_{ijk} \leftarrow r_k$.bound $\cap\, b_{ij}$
11: $b_i \leftarrow b_i - b_{ijk}$
12: r_k.bound $\leftarrow r_k$.bound $- b_{ijk}$
13: **if** r_k.bound $= \emptyset$ **then**
14: DELETEREGION(r_k)
15: **end if**
16: $r_{ijk} \leftarrow$ NEWREGION(b_{ijk})
17: $r_{ijk}.\overrightarrow{\text{maps}} \ll \{m_{[i]}, m_{[j]}, m_{[r_k \cap r_{ijk}]}\}$
18: $h \ll \{r_k, r_{ijk}\}$
19: **end if**
20: **end for**
21: **end if**
22: **end for**
23: **if** $b_i \neq \emptyset$ **then**
24: $r_i \leftarrow$ NEWREGION(b_i)
25: $r_i.\overrightarrow{\text{maps}} \ll m_{[i]}$
26: $r \ll r_i$
27: **end if**
28: **end procedure**

29: **function** NEWREGION(b_n : [R], r)
30: $r_n \leftarrow \{\text{ID}(), b_n\}$
31: $r \ll r_n$
32: **for all** cells $l \in r_n$.bound **do**
33: $[R]_l \leftarrow r_n$
34: **end for**
35: **return** r_n
36: **end function**

37: **procedure** DELETEREGION(r_d : [R], r)
38: **for all** cells $l \in r_d$.bound **do**
39: $[R]_l \leftarrow r_0$
40: **end for**
41: $[R] \gg r_d$
42: delete r_d
43: **end procedure**

independent prior probabilities of occupancy at that cell, the values convolved (averaged), and the result returned. At no time during map construction or localization are these values statically represented.

To accomplish this, the global map is represented not by priors, but by region identifiers. A region is defined as an area of the global map that is uniquely covered by zero or more local maps. Before any mapping has begun, the entire map is covered by region 0 (r_0). When a map has been constructed, it is *regionalized* in the sense that it is overlayed on the global region map, and any intersections with pre-existing regions are merged into new regions. A table lookup is maintained for each region that points back to all overlapping local maps. In this way, those maps can be instantly queried for their priors, and those priors convolved.

This process is illustrated by Figure 3.8 and detailed by Algorithm 1.[1] In summary, the figure shows the installment of three local maps, divided into three major columns, each installment corresponding to one call to MapToRegions(). The third map in minor columns (3a) and (3b) represents two iterations over the for loop beginning on line 4 of the algorithm since there exist at this point two prior local maps with which the current map needs to be merged. Horizontally, the figure is broken into three major rows. The first simply shows the local maps and robot path over those maps as they exist at the call of MapToRegions(), reflected in the parameters $m_{[i]}$, the current local map, m, the collection of local maps; $[R]$, the global region map (a matrix); and r, the collection of regions. For example, the first row of column (2) shows $m_{[2]}$ overlapping $m_{[1]}$. The second row models lines 2 through 21 of the algorithm, where the boundary of the current map is

[1]The nomenclature used in the algorithm is as follows. Parameter lists are divided into two parts: the first being those parameters that are passed in as part of the procedure or function call; and the second being those parameters that are available outside the scope of the procedure, and thus not part of the formal parameter list. Atomic values (those that are not collections) are represented by a single letter with or without unbracketed subscripts, such as b_i or r_{ijk} and assigned using $\leftarrow$. Vectors or collections are represented as a whole by a single letter or an over-arrow word, in part by a bracketed index $m_{[i]}$, and inserted into or extracted from using $\ll$ and $\gg$, respectively. R-values containing multiple items are enclosed in curly braces; for example, line 18 is inserting two regions into the history vector h. Maps, which are represented as matrices, are identified by an uppercase identifier surrounded by square brackets, as in $[R]$. Finally, some atomic values are actually structures of related values. Regions are an example of this; they contain an implied identification number, a boundary, and a collection of local maps to which they correspond. These values are accessed using dot notation: r_k.bound or $r_{[i]}$.maps.

first intersected with a prior local map, and the result is iteratively intersected with the regions it overlays. For example, in the second row of column (3a), the boundary of map $m_{[3]}$ is first shown intersecting the portion of region $r_{[2]}$ from column (2) that falls within the boundary of $m_{[2]}$, the prior local map currently under consideration. It also shows below that its intersection with $r_{[3]}$ as the next region falling within the boundary of $m_{[2]}$. The third row of the figure shows the global region map $[R]$ as it exists after all the regions over the prior local map have been processed; that is, upon completion of the for loop ending on line 20. The regions are identified by shade and given with the local maps to which they point.

Formally, a map m is a conjunction of N small maps, denoted $m_{[i]}$, and annotated by a coordinate transform, denoted $\xi_{[i]}$:

$$m = \langle m_{[i]}, \xi_{[i]} \rangle_{i=1,\ldots,N} \tag{3.16}$$

$$M_{[i]} = \langle m_{[i]}, \xi_{[i]} \rangle \tag{3.17}$$

$$m = \{M_{[i]}\}_{i=1,\ldots,N} \tag{3.18}$$

The origin of each map is locally (0,0); the transform is a mapping of the i-th local map's pose $\xi_{[i]}$ to a global pose for that map, mapping its origin to a global coordinate and heading. It is because the local maps are not convolved with their pose distributions into the global map that the problems arising from using a single map are eliminated.

With the introduction of the layered maps, the perceptual model $P(o|m, \xi)$ must be modified. In order to extend this perceptual model to this map representation, [10] makes the conditional independence assumption:

$$P(M_{[i]}M_{[j]}|o, \xi) = P(M_{[i]}|o, \xi)\ P(M_{[j]}|o, \xi) \tag{3.19}$$

for all $i \neq j$. This assumption allows the extension of the perceptual model in the following manner:

$$\begin{aligned}
P(o|\xi, m) &= P(o|\xi, \bigotimes_{i=1}^{N} M_{[i]}) \\
&\overset{(a)}{=} \frac{P(\otimes_{i=1}^{N} M_{[i]}|o, \xi)\ P(o|\xi)}{P(o|\xi, \otimes_{i=1}^{N} M_{[i]})} \\
&\overset{(b)}{=} \frac{\Pi_{i=1}^{N} P(M_{[i]}|o, \xi)\ P(o|\xi)}{\Pi_{i=1}^{N}\ P(M_{[i]}|\xi)} \\
&\overset{(c)}{=} \frac{1}{P(o|\xi)^{(N-1)}} \prod_{i=1}^{N} \frac{P(M_{[i]}|o, \xi)\ P(o|\xi)}{P(M_{[i]}|\xi)} \\
&\overset{(d)}{=} \frac{1}{P(o|\xi)^{(N-1)}} \prod_{i=1}^{N} \frac{P(o|M_{[i]}, \xi)\ P(M_{[i]}|\xi)}{P(o|\xi)} \frac{P(o|\xi)}{P(M_{[i]}|\xi)} \\
&= \eta \prod_{i=1}^{N} P(o|M_{[i]}, \xi) \qquad (3.20)
\end{aligned}$$

where,

- the symbol $\bigotimes$ represents the convolution over the local maps $M_{[1..N]}$
- the derivation to (a) follows from Bayes' rule
- the derivation to (b) represents the convolution of the maps as the product of their individual probabilities
- the normalization constant $1/P(o|\xi)^{N-1}$ in (c) is independent of the map and may be removed from consideration
- the derivation to (d) follows from Bayes' rule applied to the first term $P(M_{[i]}|o, \xi)$ of (c), and
- (3.20) results from cross-cancelations in (d)

Here the expression $P(o|\xi, M_{[i]})$ is computed using the perception model described in Section 3.2.1.3. The approach computes the likelihood of observations separately in all maps, and then combines the resulting density multiplicatively [10].

3.2.3 Expectation Maximization. As was discussed in Section 2.2, there exists a high dimensionality to the mapping problem. As a robot travels, the range of possible poses and possible maps grows to a point that is computationally prohibitive to exhaustively process. EM is a statistical approach to reducing the scope of the problem in such a way that reasonable results can be achieved without the explicit consideration of all possible map configurations.

3.2.3.1 Expectation Step. In the context of localization, the expectation step describes the process by which a set of expected poses are calculated for a particular robot-reported pose, reflecting the fact that there is uncertainty in a pose and that the reported pose may not best fit the data.

Theory. In the E-step, the current best map m and the data are used to compute probabilistic estimates for the robot's position ξ^t at $t = 1, ..., T$, that is, $P(\xi^T|d, m)$, which per [43], can be expressed as the normalized product of two terms in the following manner:

$$\begin{aligned} P(\xi^t|d,m) &= P(\xi^t|o^1,...,o^t,u^t,...,o^T,m) \\ &\overset{(a)}{=} \eta_1\ P(o^1,...,o^t|\xi^t,u^t,...,o^T,m)\ P(\xi^t|u^t,...,o^T,m) \\ &\overset{(b)}{=} \eta_1\ P(o^1,...,o^t|\xi^t,m)\ P(\xi^t|u^t,...,o^T,m) \\ &\overset{(c)}{=} \eta_2\ P(\xi^t|o^1,...,o^t,m)\ P(o^1,...,o^t|m)\ P(\xi^t|u^t,...,o^T,m) \\ &= \eta_3\ \underbrace{P(\xi^t|o^1,...,o^t,m)}_{:=\alpha^t}\ \underbrace{P(\xi^t|u^t,...,o^T,m)}_{:=\beta^t} \end{aligned} \tag{3.21}$$

The normalizers n_1, n_2, and n_3 ensure the left-hand side of the equation sums up to one over all ξ^t. The derivation of 3.21 follows from (a) Bayes rule, (b) a commonly-used Markov assumption that specifies the conditional independence of future data from past data given knowledge of the current location and the map, and (c) Bayes rule under the assumption that in the absence of data, robot positions are equally likely.

Both α and β, analogous to the forward and backward functions of the Hidden Markov Model (HMM) alpha-beta algorithm [35, 36], are computed separately. While the computation of the α-values is a version of Markov localization using multiple local maps, the β-values add additional knowledge about the robot's position—not typically captured by Markov localization, but essential for revising past belief based on sensor data that was received later in time [43].

The computation of α^t is as follows, with the exception of the bound α^1, which akin to the initial state distribution, π, in [36], is 1 at the initial pose $(0, 0) : 0^o$.

$$\begin{aligned} \alpha^t &= P(\xi^t | o^1, ..., o^t, m) \\ &= \eta\ P(o^t | \xi^t, o^1, ..., u^{t-1}, m)\ P(\xi^t | o^1, ..., u^{t-1}, m) \\ &= \eta\ P(o^t | \xi^t, m)\ P(\xi^t | o^1, ..., u^{t-1}, m) \end{aligned} \tag{3.22}$$

where,

$$\begin{aligned} P(\xi^t | o^1, ..., u^{t-1}, m) &= \int P(\xi^t | u^{t-1}, \xi^{t-1})\ P(\xi^{t-1} | o^1, ..., o^{t-1}, m)\ d\xi^{t-1} \\ &= \int P(\xi^t | u^{t-1}, \xi^{t-1})\ \alpha^{t-1}\ d\xi^{t-1} \end{aligned} \tag{3.23}$$

Substituting 3.23 back into 3.22 yields a recursive rule for the computation of all α^t, which uses the data d, the model m, as well as the motion model $P(\xi' | u, \xi)$ and the perceptual model $P(\xi | o, m)$, detailed in Sections 3.2.1.2 and 3.2.1.3, respectively. This states that α at time t is equal to the normalized product of the observation model at time t multiplied by the summed product of the motion model at time t and α at time $t - 1$, with summation over the motion model at time $t - 1$.

The computation of β^t, which expresses the probability that the robot's final pose is ξ, is completely analogous to α, but backward in time. Since β^T does not depend on the data, the initial

β^T is uniformly distributed. For all others,

$$\begin{aligned}
\beta^t &= P(\xi^t|u^t, ..., o^T, m) \\
&= \int P(\xi^t|u^t, \xi^{t+1})\ P(\xi^{t+1}|o^{t+1}, ..., o^T, m)\ d\xi^{t+1} \\
&= \int P(\xi^{t+1}|u^t, \xi^t)\ P(\xi^{t+1}|o^{t+1}, ..., o^T, m)\ d\xi^{t+1} \qquad (3.24)
\end{aligned}$$

where,

$$\begin{aligned}
P(\xi^{t+1}|o^{t+1}, ..., o^T, m) &= \eta\ P(o^{t+1}|\xi^{t+1}, u^{t+1}, ..., o^T, m)\ P(\xi^{t+1}|u^{t+1}, ..., o^T, m) \\
&= \eta\ P(o^{t+1}|\xi^{t+1}, m)\ \beta^{t+1} \qquad (3.25)
\end{aligned}$$

The result of the E-step, $\alpha^t\beta^t$, is an estimate of the robot's locations at the various points in time t.

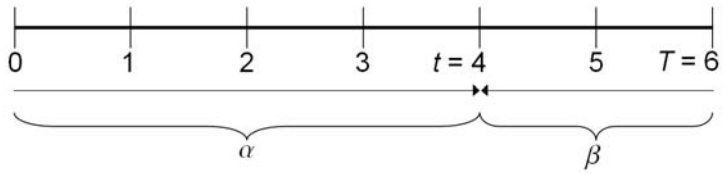


Figure 3.9: *Calculation of α and β in HMM-based EM*

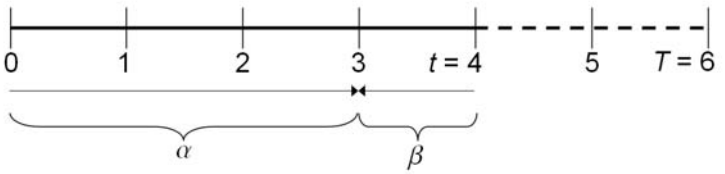


Figure 3.10: *Modified calculation of α and β in HMM-based EM*

The manner in which α and β are calculated is illustrated by Figure 3.9. T represents the total number of maps, that is, the map collected at the last iteration T. The entire map is available before the localization procedure begins. Iteration occurs from $t = 0$ to $t = T$. At each t, α is computed over $M_{0..t}$ and β is computed over $M_{t..T}$. For example, in the figure, t has progressed from $t = 0$ to

$t = 4$, and at each step, α and β have been recalculated. This process repeats at each step t, which is why this algorithm is executed off-line.

Implementation. A key difference between our implementation and the work of [10] is that between *concurrent* and *real-time* mapping and localization. The mapping procedure described in Section 4 of that article states all local maps are built *before* the EM procedure begins. This is how they are able to fully implement the β step. In contrast, a primary focus of our research is real-time localization, which limits our ability to fully implement these. Our procedure is to instead perform the forward α calculation as the map is being built, localizing the initial pose of a map at the point of that map's creation, and then to perform the backward β step over the previous iteration, as illustrated in Figure 3.10.

For both α and β, the Bayesian occupancy grid model described in section 3.1 serves to capture in large measure the utilized probabilities. That is, because the grid is built incrementally one observation at a time, and each observation builds on that of the prior observation, the probability of a given pose can be said to be conditionally dependent upon the map as it is currently represented.

The manner in which the expected pose is found is very similar to that described in [45]. In that work, the terminal end of the sonar range vectors (that is, the end at the point of the range reading) are compared against all occupied cells within the vicinity of the expected robot pose, and those for which the vector has an unobstructed path contribute to a collection of feasible poses. In this work, these vectors are aligned at their originating end, that is, at the point of the sonar, over a normally distributed set of possible sonar poses, as determined using a motion model, previously discussed.

This probabilistic motion model is computed as a Cartesian grid with the same resolution as the occupancy grid and with its origin at the reported robot position, as corrected by previous localizations. The cumulative distance and degrees of turn reported since the last localization

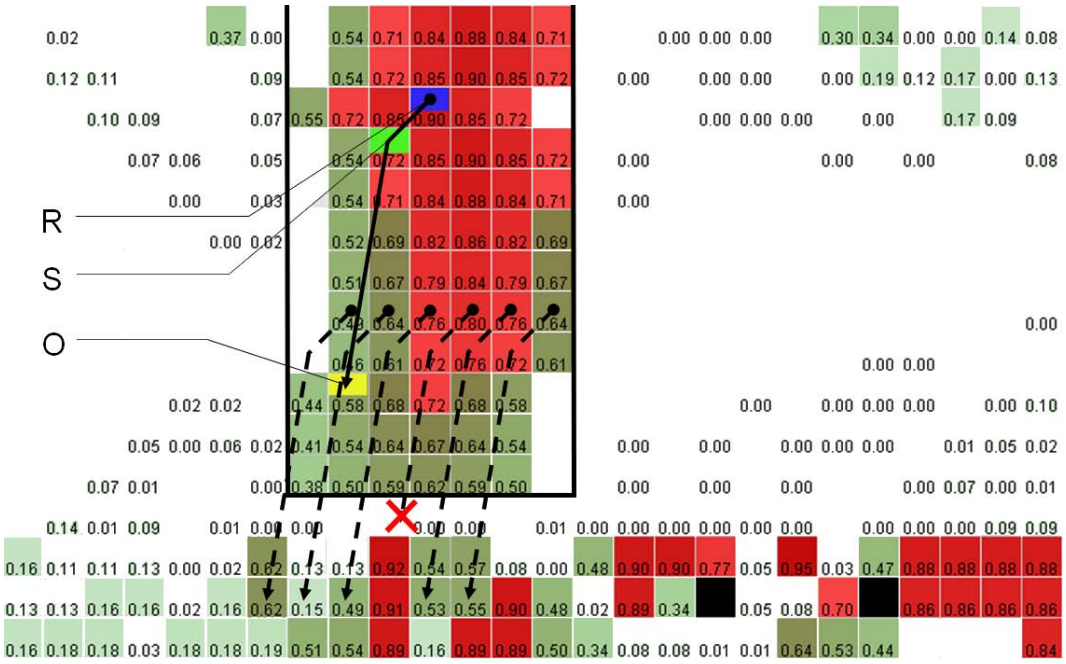


Figure 3.11: *Sample pose distribution over the motion model*

determine the dimension of this pose distribution; a greater distance increases its height, a greater degree of turn increases its width (Figure 3.6).

Figure 3.11 illustrates this procedure. The figure shows a section of a map at the point of a localization, with the motion model pose distribution overlaid on top, highlighted by the boxed region. The robot, sonar, and object positions are annotated with an R, S, and O, respectively. The pose distribution is centered on the robot pose. The vector that extends from this center to the object is mapped from every cell in the pose distribution to a prospective range reading cell. This is illustrated by the single row of mappings shown. All but one of these represents a feasible pose; the one interrupted by an X indicates a reading obstructed by the cell with a prior probability of occupancy of 0.92. If the path is unobstructed (as per the current global map), the normally-distributed probability of the cell at the terminal end of that vector being occupied is convolved with the underlying map's probability of occupancy at that same location using the Murphy model described in Section 3.1.1. This pose likelihood value is retained in a second probabilistic Cartesian

grid that is likewise aligned with the motion model over the global map at the robot's pose. These likelihood values are used in the maximization step.

3.2.3.2 *Maximization Step.*

Theory. The M-step calculates the most likely poses of the local maps given the data and current best map. Letting $t_{[i]}$ be the index of the first data point in the i-th local map, which corresponds to that map's pose, the M-Step calculates:

$$\begin{aligned} \xi_{[i]} &= \underset{\xi}{\operatorname{argmax}}\, P(\xi^{t_{[i]}} | d, m) \\ &= \underset{\xi}{\operatorname{argmax}}\, \alpha^{t_{[i]}}\, \beta^{t_{[i]}} \end{aligned} \tag{3.26}$$

Because EM is a hill climbing method, which only converges to a local maxima, if the odometric error is large, the initial map will be erroneous, and subsequent iterations of EM might not be able to recover (as in Figure 2.1 on page 7). This is extended in [10] by generating a distribution over the models that, using deterministic annealing, slowly converges to the most likely map pose; but because of our real-time constraints, this refinement is excluded.

Implementation. After all pose likelihoods have been evaluated for a sonar, a third Cartesian grid that collects histogram rather than probabilistic values is created. This grid is aligned over the global map at the reported robot pose and is updated in the following manner. Each cell that corresponds to a cell in the pose likelihood grid with a maximum likelihood value (the sonar's vote for the most likely robot position) is incremented by one. This same histogram is independently updated for each sonar. Once all sonars in a single sonar sweep are processed, those cells with the maximum value indicate all possible localized poses.

From these selected poses, those not associated with the maximum selected pose distribution are filtered out. If more than one remains, that which is closest to the original pose is selected.

Algorithm 2 *Local map localization*

```
1: procedure LOCALIZE(m_[i])
2:     [A]_ξ' ← P(ξ'|u, ξ)
3:     [C]_A ← 0
4:     for all sonars i do
5:         r_[i] ← range reading
6:         Pr(Occ) ← [Map]_i
7:         [B] ← 0
8:         for all cells j in [A] do
9:             if not obstructed(r_[ij]) then
10:                [B]_j ← [Map]_j × [A]_j
11:            end if
12:            for argmax ([B]_j) do
13:                [C]_j ← [C]_j + 1
14:            end for
15:        end for
16:    end for
17:    P_ξ' ← argmax ([C]_j × [A]_j)
18:    δ ← P_ξ − P_ξ'
19:    m_[i] ← m_[i] + δ
20: end procedure
```

At this point, a pose shift is calculated based upon the maximized pose. The maximized pose theta is represented as the delta between the angle from the previously localized pose coordinate to the current pose coordinate, and the angle from the same previous pose coordinate to the newly maximized coordinate.

3.2.3.3 Iteration. At the end of one iteration, we are left with a pose shift that is added to the accumulated pose shift calculated in previous iterations. All future robot-reported pose readings over the course of the local map are then adjusted by this accumulated shift. Once the distance of the local map has been traversed, the EM procedure is repeated for that map's initial pose using our limited implementation of the beta step described in Section 3.2.3.1. A new pose shift results, the entire map is shifted as a unit, and the accumulated pose shift for the global map is updated. At this point a new local map is initialized with a localized robot pose, the map is built, relocalized, and the process repeats. This EM localization procedure is given in Algorithm 2.

IV. Results and Analysis

This chapter presents the various results of applying the mapping and localization techniques discussed in the previous chapter, along with the many factors which influence them, such as sensors, environment, models, and parameters.

4.1 Mapping

There are a number of parameters we built into the basic occupancy grid mapping algorithm that facilitated experimentation with factors that affected map quality. These parameters are not localization parameters, per se, even though their effect on the resulting map does indirectly affect localization. These secondary effects and other localization-specific parameters are discussed in section 4.1.2.

The mapping-specific parameters, along with their default values, include:

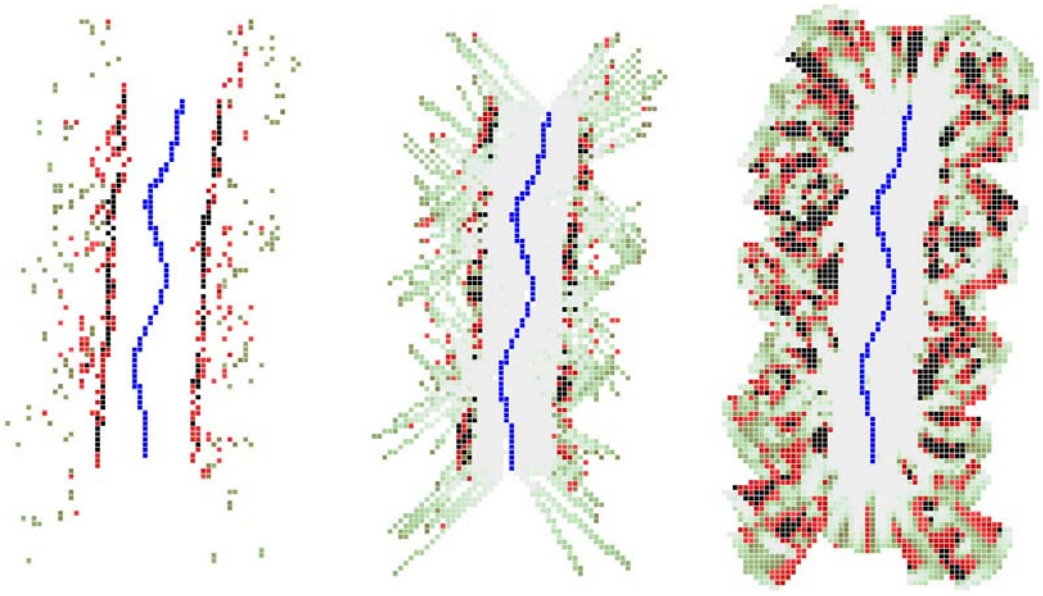
- Cell Size: the number of millimeters per grid-world cell; one sonar range reading sweep is processed per cell (100mm)
- Sonar Model: one of Point of Return, Acoustic Axis, and Field of View, as introduced in 3.1.1 (Acoustic Axis)
- Ignore Out of Range Readings: whether out of range readings are processed or ignored (Yes)
- Obstruction value: the minimum prior probability of occupancy that identifies an obstruction along the acoustic axis of a sonar reading (0.7)
- Ignore Obstructed Readings: whether to ignore an obstruction, that is, to process the reading as if it was unobstructed (No)

Additional mapping parameters specific to the Field of View sonar model are introduced in section 4.1.1.1.

4.1.1 Sonar Model. The choice of sonar model came down to which of the three produced maps most effective for localization. While Point of Return produced visually appealing maps with minimal computation, it did not provide sufficient data. At the opposite end of the spectrum, it was our expectation that Field of View would produce the best maps in the sense that they would be the richest in data, but in actuality, they were the worst, especially in the context of integrating local maps, largely due to the fact that a particular reading has a large area of influence that often times is rather destructive, discussed in more detail below. Our localization results, then, almost entirely relied on the maps produced using the Acoustic Axis model. Figures 4.1 and 4.2 illustrate the three models, with the former processing all range readings and the latter ignoring those that are obstructed by previous readings as well as those that are out of range. For these examples, an obstruction is defined as a cell that contains a prior probability of 0.7 or greater.

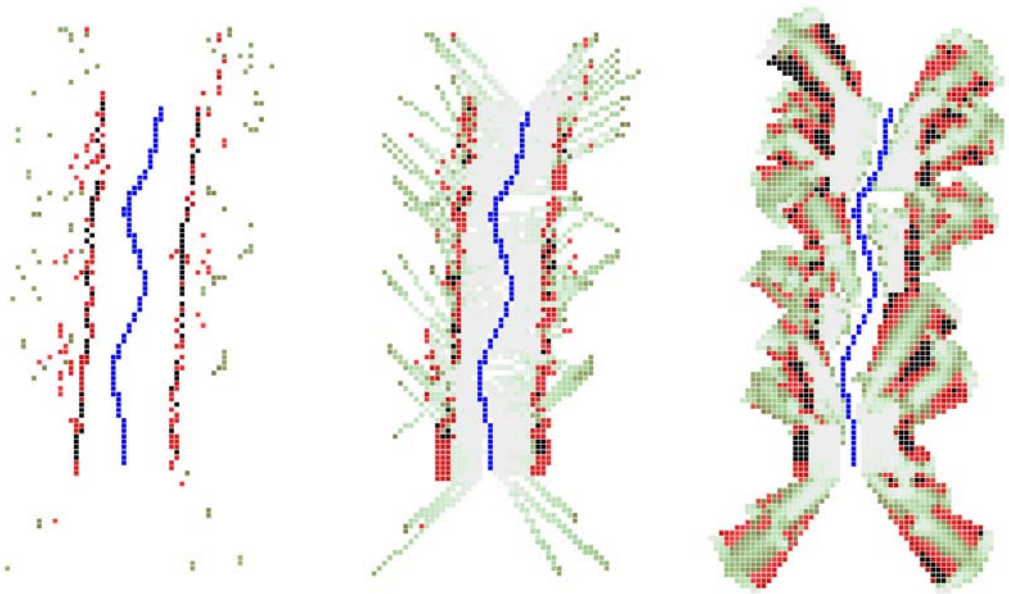
4.1.1.1 Additional Parameters. Many of the additional parameters are specific to the Field of View model, which were inserted during development in an effort to improve the quality of the map. They are shown here, with their default values, and illustrated in Figures 4.3 through 4.7.

- β: The half width of the sonar field of view (15^o)
- Region I Width: The size of Region I (200mm)
- Max P(Occupied), Region I: The cap on the maximum probability of occupied for those readings that fall within Region I (0.98)
- Max P(Occupied), Region II: A cap on the maximum probability of occupied for those readings that fall within Region II (1.00)
- Out of Range Conversion: Fixed value to which out of range readings are converted (1000mm)
- Disable Sonars: Any sonar can be selectively enabled or disabled, designed to reduce noise (all enabled)



(a) (b) (c)

Figure 4.1: *Sonar model comparison*, showing Point of Return (a), Acoustic Axis (b), and Field of View (c) models. Obstructed and out of range readings are not ignored.



(a) (b) (c)

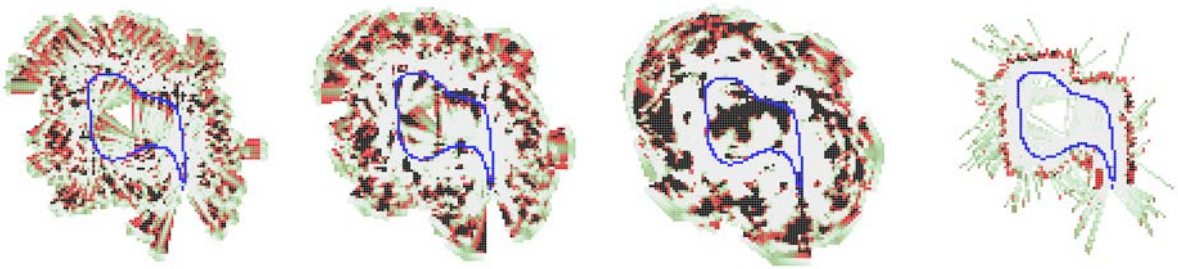
Figure 4.2: *Sonar model comparison, ignoring obstructed and out of range readings*, showing Point of Return (a), Acoustic Axis (b), and Field of View (c) models.

The conclusion to be drawn from the β configuration parameter, illustrated in Figure 4.3, is that there is little motivation for deviating from the sonar specified default of 15°. This is the value that was in fact used for all testing in our research.

Region I width was left at the default value for all testing, which when mapped to the grid world, resulting in a width of 1 to 2 cells, depending upon the heading of the range reading vector.

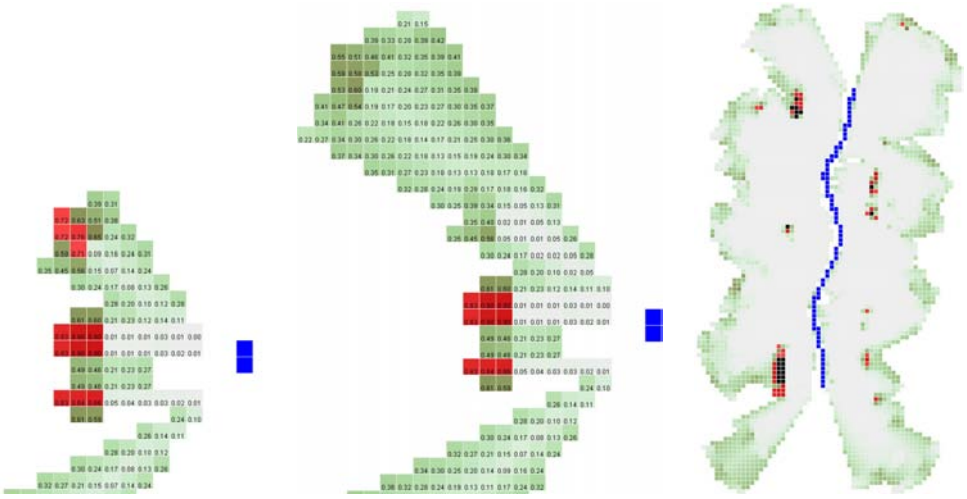
The cap on the maximum probability of occupied for Region I, which captures the notion that there is never 100% certainty of occupancy for a given cell, was also left at the default value of 0.98 for all testing.

Figures 4.4 and 4.5 illustrate the effects of capping Region II probabilities in a manner similar to that of Region I. Both map snapshots (a) and (b) in these figures show readings for a sonar sweep in progress at time $t = 2$. The large reading in (b) is seen to have written over the previous one in (a). While in this case obstructed readings are being skipped, this occurs because obstructions are checked only along the acoustic axis. These same snapshots also specify the actual prior probabilities that result from convolving the sonar reading with the existing map. The boundary between Region I and Region II is readily discernable, as well as how the two regions are inverses of each other, in accordance with the sonar model described in section 3.1.1. For Region I, cells closest to the acoustic axis receive higher probabilities of occupancy, while for Region II, there is in effect a greater certainty of vacancy.



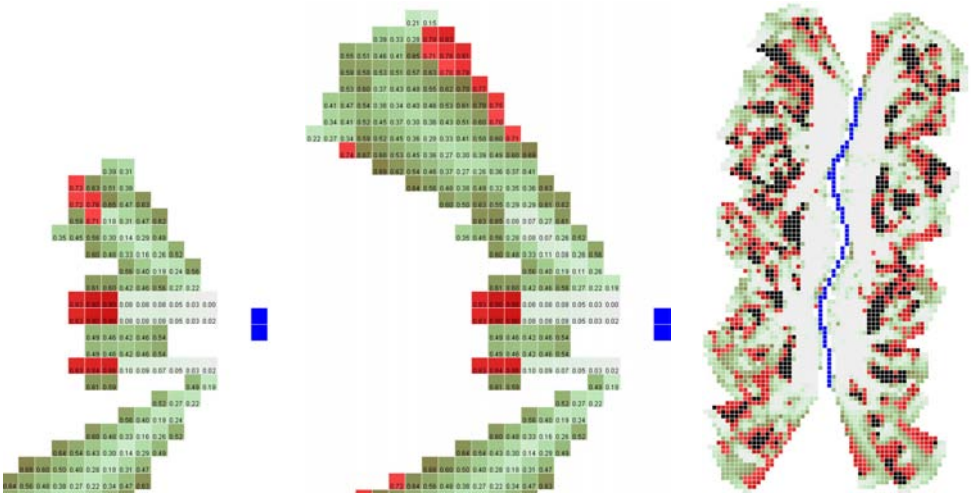
(a) (b) (c) (d)

Figure 4.3: *Field of view β configurations*, showing β values of 7° (a), 15° (b), and 30° (c). For reference, map (d) is built using the Acoustic Axis model.



(a) (b) (c)

Figure 4.4: *Field of view Region II cap of 0.5*, showing a reading in (a) subsequently overwritten by (b), resulting in final map (c).



(a) (b) (c)

Figure 4.5: *Field of view Region II cap of 1.0*, showing a reading in (a) subsequently overwritten by (b), resulting in final map (c).

The stark contrast between these two figures illustrates the effect of capping the maximum probability in Region II to 0.5 (Figure 4.4) versus leaving it at the default Murphy sonar model specification of 1.0 (Figure 4.5). It seems reasonable a cap of 0.5 would be correct, since for the environment space laying between the sonar device and the detected object, we are representing more the probability of vacancy than occupancy, and so values should fall between 0.0 (vacancy is absolutely certain) to 0.5 (there is no certainty as to vacancy or occupancy). In fact, this is how [32] implements the model. Surprisingly, however, leaving the parameter as specified by Murphy (setting it to 1.0 in effect removes the cap) and allowing values to fall between 0.0 and 1.0 produces the better result. The "burrowing" effect is much more pronounced in Figure 4.5(c), and the hallway indeed discernable; in Figure 4.4(c), the hallway is almost completely obliterated.

When out of range readings are processed for the Field of View model, one of two things can occur within our implementation: it can be ignored entirely, which means it is skipped and the next reading is processed; or it can be converted and processed as an in-range reading. These options are illustrated in Figure 4.6, where three different conversions are shown along with representations

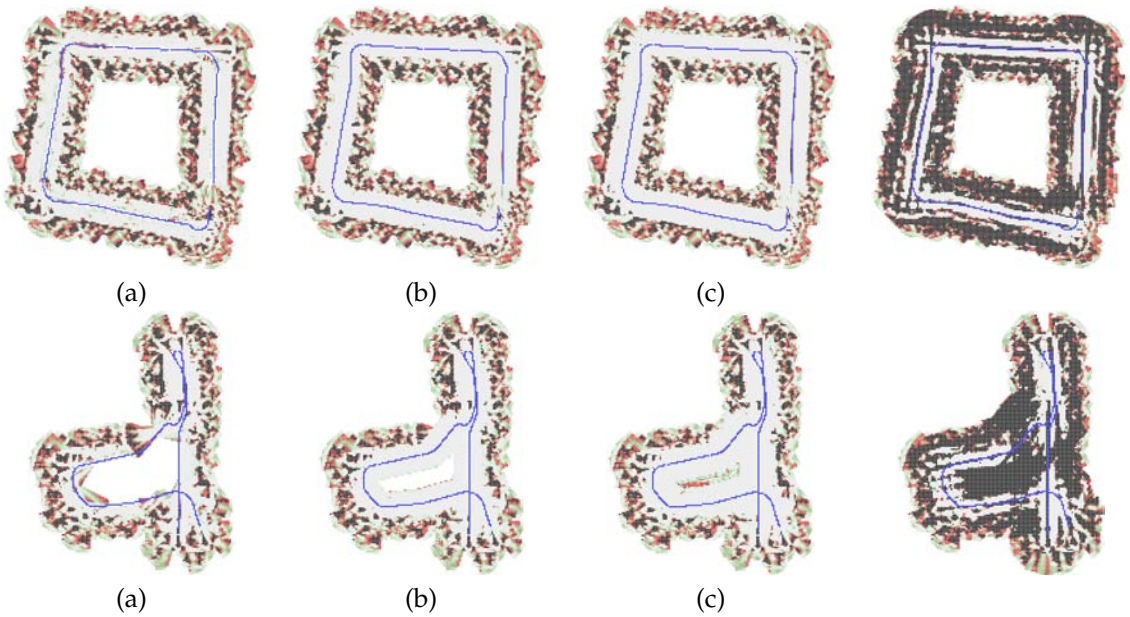
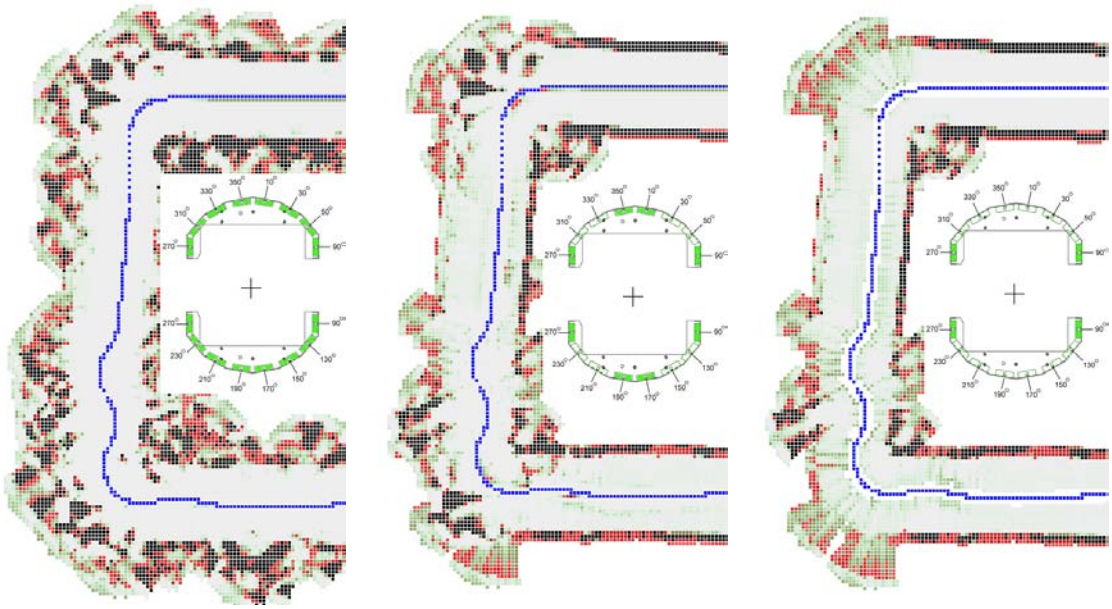


Figure 4.6: *Range reading configurations*, showing out of range readings ignored (a) or converted to 1000mm (b), 1500mm (c), and 2500mm (d).

where the out of range readings are ignored. The maximum range of the sonar for the Pioneer is 3000mm, so the 2500mm option is shown less for utility and more for its destructive effect. For the upper map, the 1500mm setting produced the cleanest map, although only marginally over the 1000mm setting (as can be seen by comparing the lower-right corners). For the lower map, the 1000mm setting was best, which is more clearly seen in that it virtually eliminated the phantom island that appeared using higher conversion values. The key difference between these two environments is the upper map is a hallway with no open areas; the lower map has a large open space that better demonstrates the spurious Field of View readings that occur in such a configuration. This suggests that while 1500mm may be best in the absence of open spaces, the 1000mm settings works almost equally well, and has the advantage of handling open spaces better.

The final parameter configuration within our implementation that is most applicable to the Field of View model (although it affects the others as well), is that of enabling and disabling specific sonars, illustrated in Figure 4.7. For the maps shown, out of range readings are converted



(a) (b) (c)

Figure 4.7: *Sonar configurations*, showing effects of enabling certain sonars, as indicated by the inset sonar array diagrams (where shading indicates an enabled sonar).

to 1000mm, and obstructed readings are not ignored. This parameter highlights the noisy nature of sonars, especially when considering those that strike a wall in a non-orthogonal manner. In this environment and with this model, however, the extra noise serves to clean up path (a), largely as a result of the out of range reading conversion; the other configurations each leave the path with more residue, although walls generally end up being more well defined (as seen in the upper and lower hallways). Because the Field of View model is best considered as a "burrowing" mapper, the maps with the cleaner walls are actually inferior to (a).

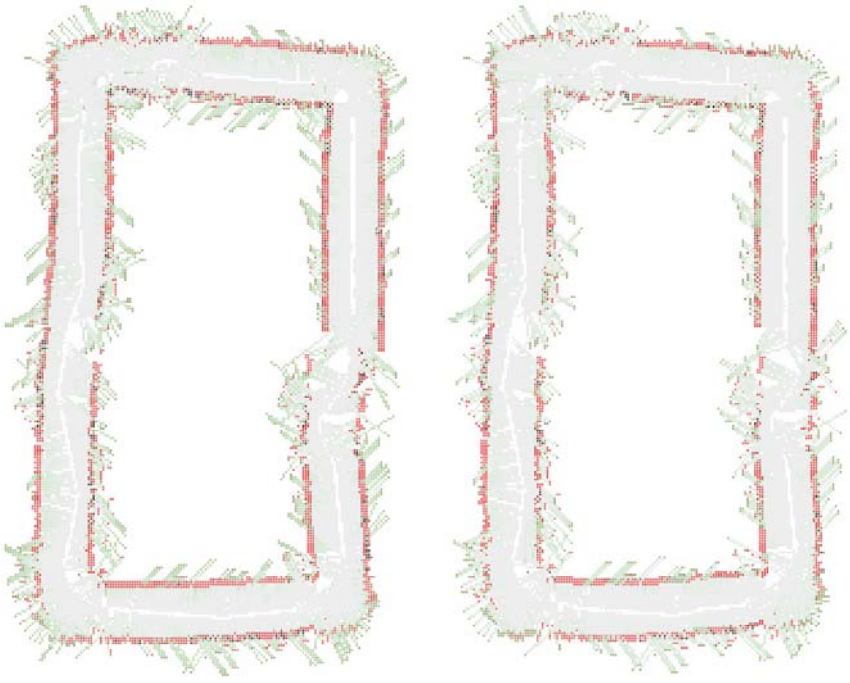
4.1.2 Localization. The goal of this research effort was to achieve a form of *local* localization, that is, localization within a single room, that would contribute to a higher level localization between rooms. Our initial testing was done in the context of reconciling pose shift that occurs when mapping cycles, such as hallways. In several cases these results were good, as demonstrated in Figures 4.8 through 4.10. A negative result, which is characteristic of attempting to localize in large environments, is shown in Figure 4.11.

Figure 4.8 shows the bottom half of a corridor in the shape of a figure 8 that spans roughly 44m. The upper left and right corners show openings that correspond to the hallways that extend upward in that direction. For this simulation run, the local map size was 5m, out of range readings were skipped, and cells with probability of occupancy 0.7 or higher were considered as obstructions. The improvement in the map as a result of our localization is most readily seen by looking at the lower right corner where the robot returns to its starting position and closes the loop. In this case, you can see the gap is smaller.

The second example, Figure 4.9, shows the robot mapping the same environment as that in Figure 4.8, but in this case, the path circles the entire corridor, rather than just the lower half, about 66m in length. The openings to the hallway that connects the east and west corridors are visible on the interior wall in the middle of the map. The local map size was 5m, out of range readings were skipped, and cells with probability of occupancy 0.7 or higher were considered as obstructions.

(a) (b)

Figure 4.8: *Localization result A*

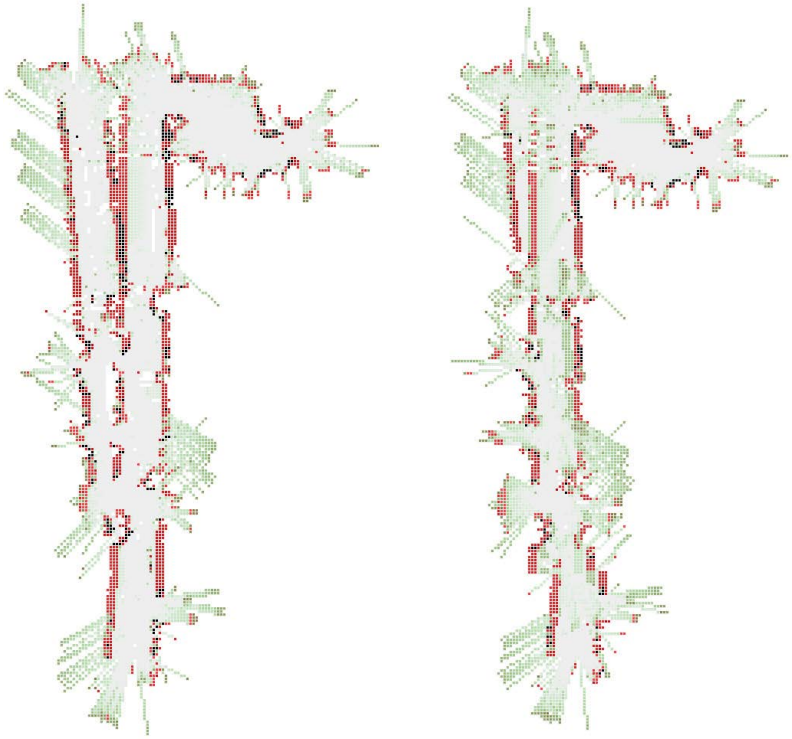


(a) (b)

Figure 4.9: *Localization result B*

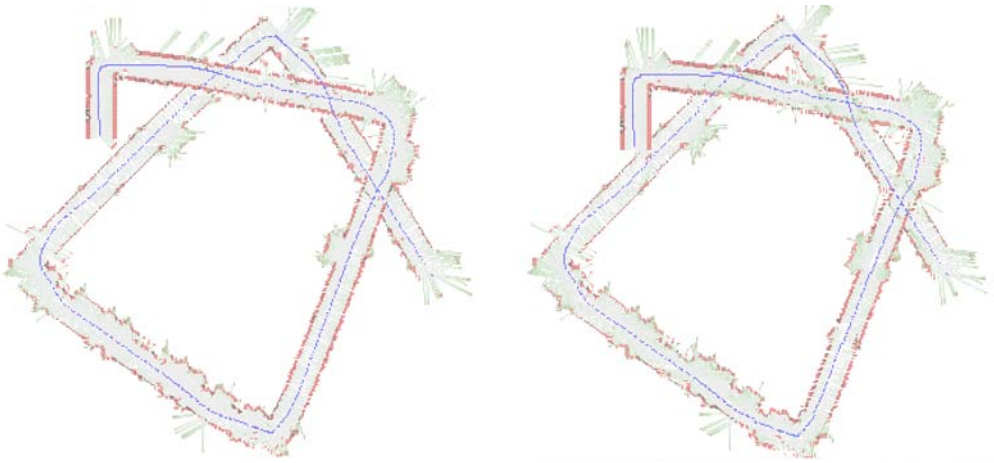
Again, the correction resulting from localization is apparent when focusing on the closure of the cycle, where the misalignment in the original map (a) is completely aligned in (b).

Figure 4.10 shows a real-world example of improved localization. In this test, the robot start position was at the upper right-hand corner, and its path extended forward a short distance, left around a 90º turn, down a hallway with doorways and obstacles along the wall, and after a 180º turn, back to the start position, for a total distance traveled of approximately 50m. The settings were the same as for the two runs just discussed. The original map (a) shows plainly the odometry error that occurred after making the 180º turn. In the localized map (b), there is some correction, but the pose of the local map created as the robot approached the 90º right turn back to its starting position was not corrected sufficiently. This is because the drift was great enough that the original wall ended up in the center of the new path and the localization procedure was unable to compensate.



(a) (b)

Figure 4.10: *Localization result C*



(a) (b)

Figure 4.11: *Localization result D*

The correspondence problem is magnified in the large cycle of Figure 4.11, which covers approximately 120m of distance. For this real-world run, the robot's starting position is seen at the right side of the figure. It's path extends up toward the top of the figure, makes a 90^o turn down toward the left side, and after two more turns, makes its way back to the starting position (which it sorely misses), and then revisits the initial corridor, and stops after one final turn (shown in the upper left-hand corner of the figure). This example highlights the problem that occurs when the robot does not soon return to a portion of the environment is has previously visited. Another factor is the extent of the odometry error, which becomes significant at the second turn (shown at the lower left-hand corner of the figure). Our localization algorithm is unable to make any correction to this map.

Our final example (Figure 4.12) illustrates an environment that is in contrast to the last, in that it is a room that spans a much shorter distance (being approximately 14m long by 6m wide). In this case, we ignored obstructions, and the local map size was set to 3m. The robot path started in the doorway shown in the lower left-hand corner of (a), made a circle around and loop within the room, exited through the doorway (b), traveled the adjacent hallway, reentered the room through a second doorway, and made a final trip around the room back to its starting point. Figures (a)

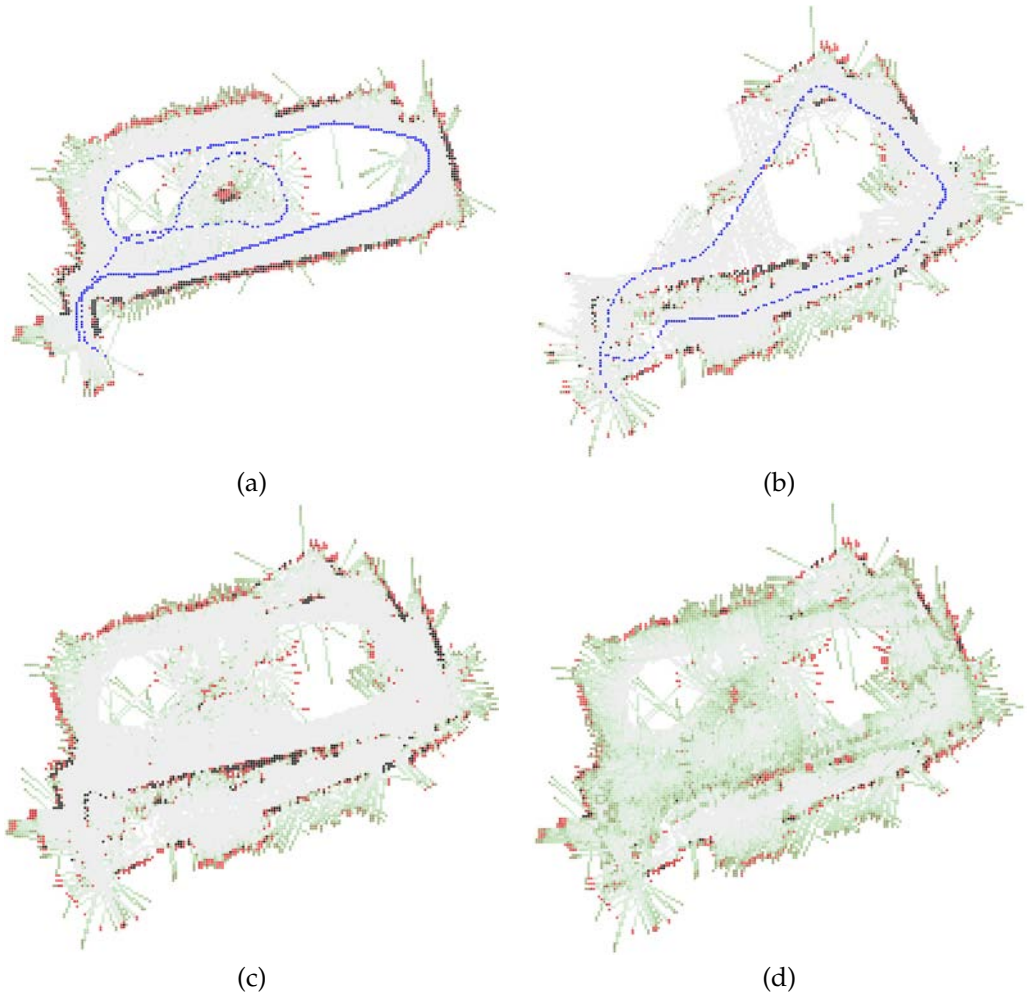


Figure 4.12: *Localization result E*

and (b) show the single run in two halves. Figure (c) shows the resulting unlocalized map, and (d) our attempt at localization. The first leg of the mapping cycle (a) shows very little odometry error, even with a large number of turns and a loop around an obstacle in the room's center. It is not until leaving the room and making a sharp 90° turn to the left that any noticeable error occurs. Again, it's significant enough that our procedure is unable to correct for it, and once the run is complete, we are left with a map not much different from the original, with the exception of significant additional noise.

It is our belief that one contributing factor to our poor results was that the pursuit of real-time localization limited our implementation of the alpha-beta algorithm (Section 3.2.3) to such a degree as to not be consistently reliable. Rather than relocalize all local maps at each iteration, we localize the current map at two points: at its creation (alpha), and at its completion (beta), both from its point of origin.

Another factor is the *local* aspect of our localization. Localization only occurs at one point for each local map (at two different times, as just discussed). The pose distribution, that is, the region considered for possible pose correction, only composes a small section of the world. As a result, when the robot is traveling over large areas, our implementation has no knowledge of large scale drift, as evidenced by Figure 4.11. For smaller environments, like the one shown in Figure 4.12, the prior knowledge is confined. Unfortunately the improvement seen in this particular example is minimal.

Our conclusion is the implementation shows promise, but needs to be extended and refined, as discussed in Chapter 5.

V. Future Work and Conclusions

This chapter discusses a number of extensions that we feel would result in improved localization results, and then presents concluding remarks regarding our research effort.

5.1 Future Extensions

One of our localization steps may be contributing to the noise of our resulting maps. Specifically, knowledge of the environment is only partial for the localization step that occurs at the creation of a local map. That is, only the environment to the rear has been explored, and that to the front remains unexplored, with the exception of those instances when the robot is revisiting a previously explored area of the environment. Additionally, the localization is occurring using sonar data acquired at a position that is up to 100mm further along the path than that at which the data it is localizing against was acquired. An alternative to this new map localization is to slide our two localization steps back one time cycle; that is, localize the maps at $t-2$ and $t-1$ rather than t and $t-1$. This will provide the localization algorithm with more complete environment data from which it can better maximize the expected pose.

As concerns localization of the map as a whole, it would be worthwhile to expand the beta step such that it traverses further back in time than the one step we presently have. Ideally, a full traversal back to $t = 0$ would be implemented, but the real-time constraints prohibit this. It is reasonable that the extent of the beta calculation could be dynamic in the sense that as time allows, the calculation continues. This would accommodate the dynamic nature of the environment, where some localization steps take less time to process than others by virtue of limited sonar data or a smaller local map size.

Our expectation calculation uses independent sonar voting (Section 3.2.3.2), meaning, each sonar has a vote on which robot pose within the pose distribution best fits its reading given the map, and each sonar is considered independently of the others. In other words, each sonar updates

the pose histogram without consideration for any of the other readings. A more robust approach would be to combine the votes of all sonars at each point in the pose distribution so that the pose that simultaneously best fits all the readings would be selected. This would not cost any more computationally because the same number of calculations would be performed, and would lead to a better-informed expectation calculation.

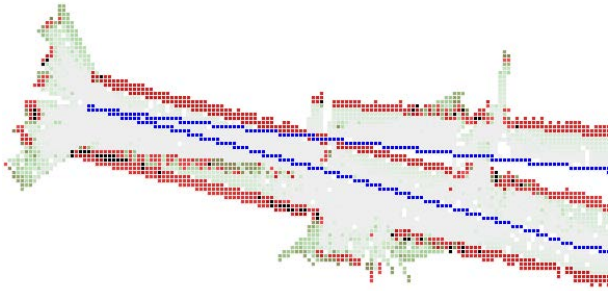


Figure 5.1: *Map segment with unrecoverable pose drift*

In certain environments, especially those where the robot is visiting areas it has previously been, it may be worthwhile to localize using the global map as it existed prior to the construction of the current local map. For example, Figure 5.1 shows an instance where doing so may correct for the pose drift that occurred when the robot reached the end of the hall, turned approximately 180^o, and proceeded back down the same path. Consider the return path shown to be mapped over two local maps. If at the end of the first the robot's reported pose is compared with the global map *before* that map is convolved, it would have a greater likelihood of determining it is on the wrong side of the wall and correct for it. In our implementation, the local map is convolved first, when in a scenario as this with such a large drift, the ability to recover quickly becomes impossible.

Two additional extensions involve the collection of sonar range data. The first is to change the frequency with which sonar sweeps are processed for a given cell. In our implementation, we process one sweep. While processing multiple sweeps while the robot is stationary would likely yield little change, doing so while it moves over the course of a single cell may see a reduction in noise that would be beneficial to cleaning the maps. Another means of cleaning the sonar data

would be to take those readings that can be identified as spurious when compared to adjacent and prior readings and modify them such that they strengthen those other readings.

Another facet of this involves the fact that in our implementation, if the robot's position is not changing, sonar data is not collected, even if it is turning. As a means of acquiring greater knowledge, it may be worthwhile to not only incorporate sonar data as the robot moves, but also as it turns, which would be most beneficial around corners where the robot is most likely to slow down and possibly turn several if not many degrees over the course of 100mm.

5.2 Conclusion

The objective of this research effort was to explore and develop a real-time sonar-based robot mapping and localization algorithm that provides pose correction within the context of a single room, to be combined with pre-existing global localization techniques, and thus produce a single, well-formed map of an unknown environment. The particular direction chosen was based on the work of [10, 43, 45], specifically in the context of Baum-Welch-based expected pose maximization. Our algorithm implemented a reduced form of the alpha-beta algorithm, performing the forward alpha calculation as an integral component of the occupancy grid mapping procedure using local maps, and a backward beta calculation that considered the map at $t - 1$ only.

Real-time localization is an extremely difficult task that continues to be the focus of much research in the field of autonomous robotic mapping. Most advances in localization have been achieved in an off-line context. The results of our research and implementation did not produce the desired localization in a consistent number of cases, a factor we believe is most directly attributable to the limited beta calculation and the limited knowledge utilized by the algorithm at each localization step. Unfortunately, the processing requirements to perform a full beta calculation cannot be accomplished in real time with current computer resources. However, we believe there

is ample room for extension of this particular research, as outlined above, that may well lead to a successful real-time local localization algorithm.

Bibliography

1. Abrash, Michael. "Of songs, taxes, and the simplicity of complex polygons". *Dr. Dobb's Journal*, 16(6):139–144, June 1991.
2. Aekaterinidis, J., K. Kostoulakis, L. Doitsidis, K. P. Valavanis, and N.C. Tsourveloudis. "An Interface System for Real-Time Mobile Robot Environment Mapping using Sonar Sensors".
3. Avots, D., E. Lim, R. Thibaux, and S. Thrun. "A probabilistic technique for simultaneous localization and door state estimation with mobile robots in dynamic environments", 2002.
4. Biswas, R., B. Limketkai, S. Sanner, and S. Thrun. "Towards object mapping in dynamic environments with mobile robots", 2002.
5. Borenstein, J. and B. Everett. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Wellesley, MA, 1996.
6. Borenstein, J. and Y. Koren. "Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance". *IEEE Journal of Robotics and Automation*, volume 7.
7. Borenstein, J. and Y. Koren. "The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots". *IEEE Journal of Robotics and Automation*, volume 7, 278–288. 1991.
8. Brown, R. G. and B. R. Donald. "Mobile Robot Self-Localization without Excplict Landmarks". *Algorithmica*, 26. 2000.
9. Burgard, W., D. Fox, H. Jans, C. Matenar, and S. Thrun. "Sonar-based Mapping of Large-Scale Mobile Robot Environments Using EM". *Proceedings of the International Conference on Machine Learning*. Bled, Slovenia, 1999.
10. Burgard, W., D. Fox, H. Jans, C. Matenar, and S. Thrun. "Sonar-Based Mapping With Mobile Robots Using EM". *International Conference on Machine Learning*. 1999.
11. Castellanos, J. and J. Tardos. *Mobile Robot Localization and Map Building*. Kluwer Academic Publishers, Boston, MA, 2000.
12. Csorba, M. *Simultaneous Localisation and Map Building*. Ph.D. thesis, University of Oxford, 1997.
13. Dellaert, F., S. Seitz, C. Thorpe, and S. Thrun. "EM, MCMC, and chain flipping for structure from motion with unknown correspondence", 2000.
14. Dempster, A., A. Laird, and D. Rubin. "Maximum likelihood from incomplete data via the EM algorithm", 1977.
15. Dissanayake, G., H. Durrant-Whyte, and T. Bailey. "A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem". *Mobile Robot Navigation and Mapping, Working notes of ICRA 2000 Workshop W4*. April 2000.
16. Dissanayake, G., P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba. "A solution to simultaneous localisation an map building (SLAM) problem". *IEEE International Conference on Robotics and Automation*. 2001.
17. Doiucet, A., N. de Freitas, K. Murphy, and S. Russell. "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks". *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 176–183. 2000.
18. Durrant-Whyte, H., S. Majumder, S. Thrun, de Battista, and S. Scheding. "A Bayesian algorithm for simultaneous localization and map building". *In Proceedings of the 10th International Symposium on Robotics Research (ISRR '01)*. Lorne, Australia, 2001.

19. Elfes, A. "Sonar-based real-world mapping and navigation". *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, June 1987.

20. Elfes, A. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. Ph.D. thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.

21. Fox, D., W. Burgard, and S. Thrun. "Markov Localization for Mobiel Robots in Dynamic Environments".

22. Gutmann, J.-S. and K. Konolige. "Incremental Mapping of Large Cyclic Environments". *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*. 2000.

23. Iocchi, L., K. Konolige, and M. Bajracharya. "Visually realistic mapping of a planar environment with stereo". *In Proceedings of the 2000 International Symposium on Experimental Robots*. Waikiki, Hawaii, 2000.

24. Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems". *Trans. ASME, Journal of Basic Engineering*, 82, 35–45. 1960.

25. Laviers, Kennard R. *Concurrent Cognitive Mapping and Localization using Expectation Maximization*. Master's thesis, Air Force Institute of Technology, 2004.

26. Leonard, J. and H. Feder. "A computationally efficient method for large-scale concurrent mapping and localization". J. Hollerbach and D. Koditschek (editors), *Proceedings of the Ninth International Symposium on Robotics Research Robotics Research*. Salt Lake City, Utah, 1999.

27. Lu, F. and E. Milios. "Globally consistent range scan alignment for environmental mapping". *Autonomous Robots*, 4:333–349, 1997.

28. Lui, Y., R. Emery, D. Chakrabarti, W. Bugard, and S. Thrun. "Using EM to learn 3D models with mobile robots". *In Proceedings of the International Conference on Machine Learning (ICML)*. 2001.

29. Martin, C. and S. Thrun. "Online acquisition of compact volumetric maps with mobile robots". *IEEE International Conference on Robotics and Automation (ICRA)*. Washington, DC, 2002.

30. McLachlan, G. and T. Krishnan. *The EM Algorithm Extension*. Wiley Series in Probability and Statistics, New York, 1997.

31. Montemerlo, Michael. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.

32. Moravec, H. "Sensor fusion in certainty grids for mobile robots", 1988.

33. Murphy, Robin R. *An Introduction to AI Robotics*. MIT Press, Cambridge, Massachusetts, 2000.

34. Newman, P. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*. Ph.D. thesis, Australian Centre for Field Robotics, University of Sydney, Sydney, Australia, 2000.

35. Rabiner, L. R. and B. H. Juang. "An Introduction to Hidden Markov Models". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 12, 4–16. January 1986.

36. Rabiner, Lawrence R. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". *Proceedings of the IEEE*, volume 77, 257–286. February 1989.

37. Shatkay, H. *Learning Models for Robot Navigation*. Ph.D. thesis, Brown University, Providence, RI, 1998.

38. Smith, R. and P. Cheeseman. *On the representation of spatial uncertainty*. Technical Report 4760 and 7239, SRI, 1985.

39. Smith, R., M. Self, and P. Cheeseman. "Estimating uncertain spatial relationships in robotics". *Autonomous Robot Vehicles*, 167–193, 1990.

40. Thrun, S. "Exploration and Model Building in Mobile Robot Domains". E. Ruspini (editor), *Proceedings of the IEEE International Conference on Neural Networks*, 175–180. IEEE Neural Network Council, San Francisco, CA, 1993.

41. Thrun, S. "A Probabilistic Online Mapping Algorithm for Teams of Mobile Robots". *International Journal of Robotics Research*, 19(11):972–999, 2001.

42. Thrun, S., W. Burgard, and D. Fox. "A Real-Time Algorithm for Mobile Robot Mapping with Applications to Multi-Robot and 3D Mapping". *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2000.

43. Thrun, S., D. Fox, and W. Burgard. "A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots". *Machine Learning*, 31:29–53, 1998.

44. Thrun, Sebastian. *Robotic Mapping: A Survey*. Morgan Kaufmann, 2002.

45. Varveropoulos, Vassilis. "Robot Localization and Map Construction Using Sonar Data". *The Rossum Project*.

46. Wikimedia Foundation. "Bresenham's Line Algorithm". Sep 2004. URL `http://en.wikipedia.org/wiki/Bresenham's_line_algorithm`.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
21-03-2005	Master's Thesis	Jan 2004 – Mar 2005

4. TITLE AND SUBTITLE

Robot Mapping With Real-Time Incremental Localization Using Expectation Maximization

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Owens, Kevin L., Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCS/ENG/05-12

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Dr. Mikel M. Miller AFRL/SNRP
2241 Avionics Circle
WPAFB OH 45433-7333
(937) 255-6127 ext. 4274

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research effort explores and develops a real-time sonar-based robot mapping and localization algorithm that provides pose correction within the context of a singe room, to be combined with pre-existing global localization techniques, and thus produce a single, well-formed map of an unknown environment. Our algorithm implements an expectation maximization algorithm that is based on the notion of the alpha-beta functions of a Hidden Markov Model. It performs a forward alpha calculation as an integral component of the occupancy grid mapping procedure using local maps in place of a single global map, and a backward beta calculation that considers the prior local map, a limited step that enables real-time processing.

Real-time localization is an extremely difficult task that continues to be the focus of much research in the field, and most advances in localization have been achieved in an off-line context. The results of our research into and implementation of real-time localization showed limited success, generating improved maps in a number of cases, but not all—a trade-off between real-time and off-line processing. However, we believe there is ample room for extension to our approach that promises a more consistently successful real-time localization algorithm.

15. SUBJECT TERMS
Robot mapping, real-time mapping and localization, expectation maximization, Bayesian occupancy grid map, local maps

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
REPORT	ABSTRACT	c. THIS PAGE			Gilbert L. Peterson, Dr., USAF (ENG)
U	U	U	UU	78	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4281